2016

3D Perception Based Lifelong Navigation of Service Robots in Dynamic Environments

Dylan Schwesinger
Lehigh University

3D Perception Based Lifelong Navigation of Service Robots in Dynamic Environments

by

Dylan Schwesinger

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Science

Lehigh University
September 2016

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dylan Schwesinger
3D Perception Based Lifelong Navigation of Service Robots in Dynamic Environments

---

Date

John Spletzer, Dissertation Director, Chair

Accepted Date

Committee Members

Brian Chen

Xiaolei Huang

Jason Derenick

Contents

List of Tables

List of Figures

Abstract

Lifelong navigation of mobile robots is to ability to reliably operate over extended periods of time in dynamically changing environments. Historically, computational capacity and sensor capability have been the constraining factors to the richness of the internal representation of the environment that a mobile robot could use for navigation tasks [1, 2]. With affordable contemporary sensing technology available that provides rich 3D information of the environment and increased computational power, we can increasingly make use of more semantic environmental information in navigation related tasks [3].

A navigation system has many subsystems that must operate in real time competing for computation resources in such as the perception, localization, and path planning systems. The main thesis proposed in this work is that we can utilize 3D information from the environment in our systems to increase navigational robustness without making trade-offs in any of the real time subsystems. To support these claims, this dissertation presents robust, real world 3D perception based navigation systems in the domains of indoor doorway detection and traversal, sidewalk-level outdoor navigation in urban environments, and global localization in large scale indoor warehouse environments.

The discussion of these systems includes methods of 3D point cloud based object detection to find respective objects of semantic interest for the given navigation tasks as well as the use of 3D information in the navigational systems for purposes such as localization and dynamic obstacle avoidance. Experimental results for each of these applications demonstrate the effectiveness of the techniques for robust long term autonomous operation.

Chapter 1

Introduction

Autonomous navigation has reached increased public awareness since Google's self-driving car project [4] was spawned. This project has its roots in the 2007 DARPA Urban Challenge, the third driverless car competition where teams competed in a cluttered urban environment. Five of the six teams that finished the race: Tartan Racing [5], Stanford Racing [6], Victor Tango [7], MIT [8], The Ben Franklin Racing Team [9], and Cornell [10] all made use of the Velodyne HDL-64E LIDAR as the primary sensor for terrain map construction and obstacle detection. This sensor has the capability of returning 3D scans of the environment and is the same sensor used on the first generation Google self-driving cars.

This dissertation explores lifelong navigation for service robots using 3D sensors that are smaller and more affordable than those used on autonomous cars. The goal of lifelong navigation for mobile robots is to reliably operate over extended periods of time in dynamically changing environments. A key prerequisite of navigation systems is an internal representation of the environment. Historically, computational capacity and sensor capability have been the constraining factors in the choice of map representation [1]. With the advent of inexpensive sensors that provide rich 3D information of the environment, such as the Microsoft Kinect, and increased computational power, we are in the midst of a paradigm shift in the field of robot navigation for service robots.

Leveraging contemporary technology, the thesis proposed here is that robust lifelong navigation is enabled by:

1. 3D information from the environment,

2. a navigation system that can adapt to environmental changes.

The main contribution of this dissertation is the demonstration of robust long term autonomous navigation solutions in applied real-world settings. The area of applied mobile service robots draws on many subfields of robotics research to enable successful operation. Some of the subfields include: perception, navigation, and control. Integrating the many systems involved into a functional robot is part of the challenge of applied mobile robotics. The perception and navigation subsystems, and the integration of these systems, are the main focus of this work.

1.1 System Considerations

One important consideration in mobile robotics is the choice hardware as this strongly influences algorithmic decisions across all subsystems involved. For the applications in this work we are focused on *service* robots and specifically in this dissertation the main target platform is an electric powered wheelchair. A useful mobile robot must be able to perceive the world to some extent. Perception typically falls into two categories: proprioception, sensing of one's own internal status, and exteroception, sensing of the outside world. The proprioceptive sensors of the wheelchair platform include wheel encoders, which measure the revolution of each wheel, and an inertial measurement unit, which measures the linear acceleration and angular velocity of the wheelchair.

The exteroception of the system is the focus of the 3D perception aspect of the research. The system has various sensors available for this purpose including: the IFM O3D200, Microsoft Kinect, and Velodyne VLP-16. The commonality of all these sensors is that range information is returned from discrete samples of 3D surfaces. This information can be represented as a point cloud, a set of 3D points in some coordinate system. For the applications discussed in in this dissertation, we assume that the information that the system receives from the environment is in the form of a stream of point clouds. Furthermore, the focus of the perception subsystem is to extract salient objects from these point clouds.

1.2 Point Cloud Based Object Detection

Object detection is the process of finding instances of semantic objects of certain classes, such as pedestrians or cars. The process of detecting objects in a point cloud

involves finding subsets of the points that belong to the objects of interest.

Point cloud object detection methods can be roughly divided into three categories:

1. Assume that separate point clouds representing entire objects have been extracted from the original point cloud data. The goal is to classify the object that a point cloud represents.

2. Label a scene directly into regions belonging to object classes.

3. Perform a targeted segmentation of the point cloud data. In this case, the class of interest is known and the segmentation method is specifically designed to find objects of this class.

In the first type of method, the segmentation of objects of interest from data is a crucial preprocessing step. Nguyen and Le [11] provide a recent survey of general point cloud segmentation methods. Also in the literature are methods tailored to certain environments. Douillard et al. [12] assume an urban environment and consequently the existence of a ground plane and object segments are constructed from non-ground data.

After segmentation, the individual point cloud objects are then classified. This is typically done using machine learning methods. Examples of the first method that require only basic point clouds as input (some methods utilize additional information about the points, such as color) include Teichman et al. [13, 14] who use sequences of segmented objects tracked through time to classify objects as cars, pedestrians, bicyclists, or background. Endres et al. [15] use an unsupervised approach to discover object classes using Latent Dirichlet Allocation.

The second type of method labels individual points directly, which typically lie in regions belonging to objects, but separate object instances are not necessarily identified. As an example, Anguelov et al. [16] use an approach based on Markov Random Fields which uses local features computed at each point to produce a globally consistent point labeling. Triebel et al. [17] use an unsupervised approach based on Conditional Random Fields to discover multiple objects of a similar type that occur in a given 3D point cloud.

In the third type of method the segmentation is targeted toward finding a specific class of object. For example, Spinello et al. [18] focus on the detection of pedestrians. Wang et al. [19] consider the domain of autonomous driving and focus the segmentation to a set of classes, namely cars, pedestrians, and bicyclists. This third type

of object detection is the method used for our applications to detect specific objects: doorways, urban pole-like features, and vertical pallet rack supports.

1.3 Long Term Navigation

Mobile robot navigation is the ability for a robot to determine its position with respect to a frame of reference and then plan a path to a goal location. A mobile robot navigation system consists of three main capabilities: map construction and interpretation, localization, path planning. The map is an internal representation of the external environment that the robot is expected to operate within. Localization is the ability to interpret the map and determine its position and orientation with respect to a frame of reference. Path planning, also called motion planning, is the ability to take a movement task and decompose it into a sequence of discrete motions that achieve the task and satisfy movement constraints. Map building is the construction of the internal representation of the external environment the robot uses for localization and path planning. As mentioned above, our target platform is an electric powered wheelchair. This influences the scale of the navigation tasks, that is, we operate at human scale. The real-world navigation tasks presented in this dissertation are: doorway navigation, sidewalk-level urban navigation, and large-scale indoor warehouse localization.

1.4 Dissertation Contributions

As previously mentioned, the primary contribution this research is the demonstration of systems that perform long term robot navigation in dynamic environments via the integration of point cloud based object detection and robust navigation that makes use of this information. This is demonstrated by three different robot navigation systems: a doorway traversal system, a sidewalk level urban navigation system, and a large scale indoor warehouse localization system.

Regarding the doorway navigation system, CoPilot, this dissertation presents:

- A robust real-time method that detects open doorways using data from two Primesense Carmine sensors.
- A navigation system that traverses detected doorways while avoiding obstacles.

- Experimental results demonstrating the robustness of both doorway detection and doorway traversal.

With respect to the urban navigation system, this dissertation presents:

- A mapping trike platform that generates high fidelity point cloud reconstructions of the environment.
- A procedure to reduce the high fidelity point cloud reconstructions into the global locations of semantically interesting landmarks.
- A client platform capable of sidewalk-level navigation in urban environments via 3D registration of semantic landmarks.
- Experimental results demonstrating the effectiveness of the mapping platform including landmark segmentation and ovreall global map consistency.
- Experimental results demonstrating reliable long term navigation, specifically in the areas of 3D landmark segmentation, localization in the landmark map, and autonomous route navigation.

In regard to the warehouse localization system, this dissertation presents:

- An extension to the previous urban mapping and localization system to the domain of large scale indoor warehouse environments.
- Procedures to segment landmarks (vertical pallet rack supports) with the mapping platform and the client platform equipped with a sensor not used in the previous system, namely the Velodyne VLP-16.
- Experimental results demonstrating localization accuracy on the order of 2cm in the map learned with the mapping platform.
- Experimental results demonstrating the viability of using the client platform equipped with the VLP-16 sensor to learn the landmark maps.

1.5 Dissertation Outline

This dissertation is organized as follows: Chapter 2 describes some of the technologies that enable 3D point cloud based representations of the environment. These technologies include stereo vision, structured light, and time-of-flight. Additionally, the

specific sensors used in this dissertation are detailed under their respective technology sections.

Chapter 3 describes sensor calibration procedures. The first is a description of an intrinsic calibration procedure to obtain a greater than advertised effective range for Primsense based sensors. Following that, two extrinsic calibration procedures are discussed. The first is for sensors that can return 3D scans of the environment. The second is for sensors that only return a 2D scan of the environment and uses vehicular motion to obtain 3D information.

Chapter 4 discusses the approaches to navigation tasks that are common among our applications. The first section describes feature based map representations. The following section details a particle filter based approach to localization given a feature map. The final section describes the approach to path planning on global and local scales.

Chapter 5 discusses the doorway navigation system, CoPilot. The CoPilot system provides real-time detection of open doorways and autonomous traversal of detected doorways. Doorway detection is performed by first fusing the point cloud data from two Primesense sensors, then finding potential boundaries of doorways, and finally validating each pair of boundaries based on geometric constraints. Doorway traversal is performed by using the nearest validated doorway as a goal point and local planner that is capable of avoiding obstacles using 3D sensor data. Experimental results are presented where the system automatically detected and traversed 100 unique doorways with a 100% success rate.

Chapter 6 discusses the Smart Wheelchair System (SWS) for sidewalk-level urban navigation. The system is composed of a mapping platform with an accurate sensor suite and a client platform, the smart wheelchair with a less expensive sensor suite that navigates the environment based on the generated map. The mapping platform, a tricycle, is described in detail including the sensor suite and map generation process. The map includes the locations of semantically interesting landmarks, in this case pole-like urban features such as lamp posts. Then the SWS client vehicle is described including 3D perception based localization and path planning. Experimental results are presented demonstrating the effectiveness of map generation, localization, and long term navigation. The SWS achieved long term navigation, covering approximately 12km among the experiments performed.

Chapter 7 describes an extension of the SWS into the domain of large scale indoor

warehouse environments. The same mapping platform described in Chapter 6 is used to map a representative warehouse. The landmarks in this domain are the vertical pallet rack supports. The mapping procedure is described in detail. The client vehicle, the SWS, equipped with a Velodyne VLP-16 is used for localization within the map. Details of localization process including the 3D segmentation process used to detect the landmarks are discussed. Experimental results are presented demonstrating effective localization on the order of 2cm in accuracy and evidence to substantiate the viability of using the client vehicle itself to generate maps.

Finally, Chapter 8 discusses extensions of this research to related to the relationship between the sensor capability and the map representation. The first proposes richer map representations that contain more semantic information about the environment. This additional information can be utilized to enhance both localization and path planning. The next extension looks into approaches for modeling the dynamics of the environment explicitly in the map representation based on available sensor data. One aspect of this modeling is long term map maintenance, the goal of which is to automatically correct the map representation when the stable features of the environment slowly change over time. Another aspect of this modeling is dynamic obstacles, for instance pedestrians, that could be modeled and tracked to improve path planning.

Chapter 2

Sensing Technology

As mentioned in the previous chapter, the assumption for our robotic navigation applications is that the exteroceptive sensors are capable of producing 3D point clouds. This chapter describes some of the sensing technologies that have this capability. The following sections describe stereo vision, structured light, and time-of-flight based sensors. Also, the specific sensors used in this dissertation are described in their respective sensing technology sections.

2.1 Stereo Vision

Stereo vision is a passive 3D system in which two cameras are used to capture separate images of scene from different viewpoints using only ambient lighting. Triangulation methods are used to determine correspondences between pixels in each image. The relative depth of a given point in the scene can computed because the depth of each point is inversely proportional to the difference between the distances to the corresponding points and their camera centers. This information can then be used to generate a disparity map that encodes the 3D information of each point correspondence. The disparity values are inversely proportional to the to the distance that the object is away from the two cameras and directly proportional to the distance between the two cameras [20]. Figure 2.1 depicts a Stereo Labs ZED stereo camera and a representative depth image.

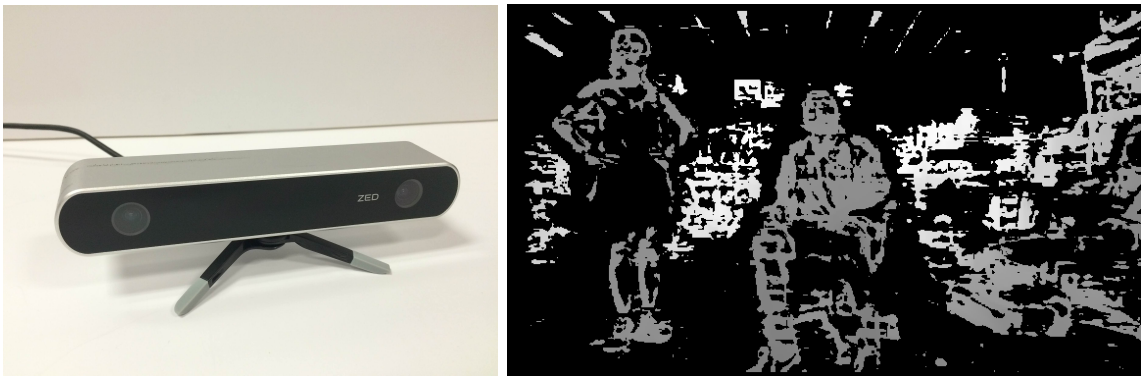

Figure 2.1: (Left) The Stereo Labs Zed stereo camera. (Right) A representative depth image captured by the Zed camera. Distance is colored on a grey to white scale where gray is nearer to the camera. Black pixels indicate that a point correspondence could not be found between the two camera images.

2.2 Structured Light

Structured light sensors work by projecting a specific light pattern on to a scene. The way that the pattern distorts when it hits a surface allows a vision system to calculate the depth information of objects in the scene. Structured light sensors use triangulation to compute disparity maps similar to the stereo vision technique, but unlike stereo vision, structured light is an active sensing technology due to the projection of the light pattern. Figure 2.2 depicts two structured light based sensors: the Microsoft Kinect and the Primesense Carmine. The later of which is used in this work.

2.2.1 Primesense Carmine

The Primesense Carmine, depicted in Figure 2.2, is a structured light based sensor using the same technology as the first generation Microsoft Kinect, but in a smaller form factor. The Carmine comes in two varieties that have different sensing ranges. The Carmine 1.08 has an advertised effective range between 80 and 350 cm and the Carmine 1.09 has a range between 35 and 140 cm. However, the calibration procedure described in Section 3.1 can be used to increase these effective ranges. The depth accuracy of the Carmine 1.08 is 1.2 cm at 2 m and the Carmine 1.09 is 0.1 cm at 50 cm. Table 2.1 contains more information about the Carmine's operational characteristics.

The main advantages of the Carmine with respect to mobile robotics applications are low cost, small form factor, light weight, and high point density. The form

Figure 2.2: Examples of structured light based sensors. (Left) The Microsoft Kinect. (Right) The Primesense Carmine 1.09.

factor and weight enable more options for mounting the sensors to the platform. Another advantage is that the Carmine streams depth images and RGB images. This enables a point cloud representation of a scene where each point is also registered to a color and this additional information can be used to aid object detection. However, the main disadvantage of the Carmine is that the structured light technology that the Carmine uses does not work in sunny outdoor environments. This limits the application domains where the Carmine could be used.

2.3 2D Scanning LIDAR

Light detection and ranging (LIDAR) sensors are based on the time-of-flight (ToF) measurement principle. Time-of-flight sensors emit a light pulse that can be reflected object's surface within the sensor's range. Based on the known speed of light, measuring the elapsed time of a light signal between its emission an reception allows us to calculate the distance between the object and the sensor. Scanning LIDARs rely on a spinning mirror and laser diode to perform the time-of-flight distance calculations. As the mirror spins the laser returns are measured point by point. Figure 2.3 depicts two representative scanning LIDARs: the SICK LMS 291 and the Hokuyo UTM-30LX. The later is is used as a component in actuated sensor described below and the former is described next.

2.3.1 SICK LMS 291

The SICK LMS 291 is a ToF based scanning LIDAR. The LMS 291 has a maximum effective range of 80 meters. In our work, we use two varieties: the LMS 291-S05 and LMS 291-S14. The main difference between them is the field of view, angular resolution, return values when operating at the 75 Hz. The LMS 291-S14 can return

Figure 2.3: Examples of scanning LIDARs. (Left) The SICK LMS 291. (Right) The Hokuyo UTM-30LX.

an additional intensity value corresponding to each point which represents the amount of light reflected back to the sensor.

The main advantages of the LMS 291 in our work are its high scan rate and accuracy. These characteristics enable high quality point cloud reconstructions of the environment. The main disadvantages of the LMS 291 for use on human scale robot platforms are its weight and power consumption (20 W). Another disadvantage with regard to our work is that the LMS 291 only returns 2D scans of the environment.

2.4 3D Time-of-Flight

There are two main categories of 3D ToF based sensor technology are scanning LIDARs and flash LIDARs. The first is an extension of the 2D scanning LIDAR technology described in the previous section; these sensors add more beams allowing for multiple simultaneous scans. Flash LIDARs capture the entire scene with a single light pulse and a dedicated image sensor and produce a range image where each pixel in the image corresponds to a distance. Some flash LIDARs are capable of simultaneously measuring the distance and reflectivity of the sensed object. Scanning LIDARs that support this capability require two revolutions of the mirror to obtain this information. Also, flash LIDARs require no moving parts. Figure 2.4 depicts the Velodyne

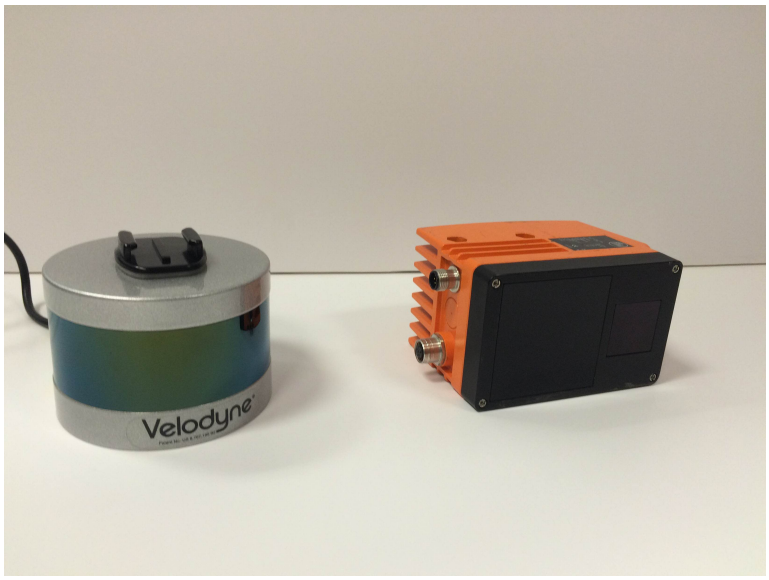

Figure 2.4: Examples of 3D LIDAR sensors. (Left) The Velodyne VLP-16 scanning LIDAR. (Right) The IFM O3D200 flash LIDAR.

Puck a 3D scanning LIDAR with 16 beams and an IFM O3D200 flash LIDAR, both of which are used in this work.

2.4.1 IFM O3D200

The IFM O3D200 is a ToF range imaging camera. While the sensor has its limitations (an effective range of $\approx$ 6 meters, a resolution of 48 $\times$ 60 pixels, a field of view of 30° $\times$ 40°, and a relatively low frame rate of $\approx$ 7 Hz), it has one critical capability. Specifically, the O3D200 can provide 3D measurements in outdoor conditions, including bright sunlight. This is something that lower-cost structured light based sensors are incapable of doing. It is also reasonably affordable, costing $\approx$\$1500 USD. At the time of the work in this dissertation, the O3D200 was an affordable sensor providing 3D sensing capability. But, the O3D200 is now obsoleted by the IFM O3D303 which provides improved sensing capability at a similar price point.

2.4.2 Actuated 3D LIDAR

This section describes an actuated Hokuyo UTM-30LX LIDAR, herein referred to the 3D Hokuyo. The motivation for actuation was to obtain 3D information from the environment to enable 3D point cloud object segmentation methods. The 3D Hokuyo

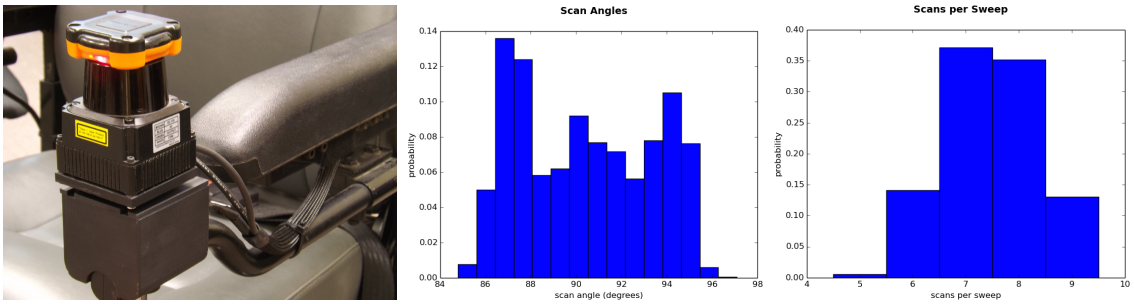

Figure 2.5: (Left) The actuated LIDAR mounted on an electric powered wheelchair. (Center) Histogram of the representative scan angles in a sweep and (Right) histogram of the number of 2D scans in a sweep. 3D aggregates of the scans in each sweep are streamed at ≈ 5 Hz.

was actuated with a Dynamixel AX-12A servo which was encased in a custom 3D printed enclosure. The 3D Hokuyo is depicted in the left image of Fig. 2.5. The entire 3D Hokuyo system has a small footprint, and is comparable in size to the standard joystick controller on an electric powered wheelchair. The actuated behavior was to simply nod up and down over a limited angular range. The Dynamixel servo is well suited for this task, as it can be controlled by explicitly setting angle commands and polled for its current angle position.

The task of the 3D Hokuyo was to aggregate the 2D laser scans from one angular set point to the next (a sweep either up or down), register each scan to a common odometric coordinate frame to account for the robot's motion, and then report the aggregation as a 3D point cloud. The design target for the 3D Hokuyo was to stream aggregated 3D scans at 5 Hz, as this was the frequency of the motion planner's control loop. A constraint on this goal was that the vertical angular resolution had to be sufficiently small in order to perform effect object segmentation. The free parameters (the choice of angular set points) were empirically determined to be $\pm 5^\circ$ from the neutral position (the LIDAR scan parallel to the ground). This choice gave us a field of view of $270^\circ \times 10^\circ$.

Fig. 2.5 shows histograms of the salient operating characteristics of the actuated LIDAR in operation. Due to factors such as inertia, the scan angles in a sweep, the number of scans in a sweep, and the amount of time a sweep takes are not deterministic. The results show that an average sweep of the actuated LIDAR contains approximately 7.5 laser scans. As the UTM-30LX scans at 40 Hz, the effective update rate of the 3D Hokuyo was 5.35 Hz, which was sufficiently close to our design target

Figure 2.6: (Top) A scan from the 3D Hokuyo. (Bottom) A photo of the corresponding scene.

of 5 Hz. Fig. 2.6 shows a visualization of the 3D point cloud data from one sweep.

2.4.3 Velodyne VLP-16

The Velodyne VLP-16, depicted in Figure 2.4 is a ToF scanning LIDAR that has 16 beams. These beams cover an vertical field of view of 30° (±15° from horizontal) with a vertical angular resolution of 2°. The horizontal field of view is 360°. The rotational speed can be set to values in the range of 5 Hz to 20 Hz corresponding to a horizontal angular resolution in the respective range of 0.1° to 0.4°. The maximum effective range of the VLP-16 is 100 meters.

The main advantages of the VLP-16 in our applications are its small footprint and 360° sensing capability. At the time of this work, the VLP-16 is a relatively recent sensing technology and largely obsoletes our work on the Actuated Hokuyo 2.4.2. The main disadvantage of the VLP-16 with respect to this work, is its low vertical angular resolution. With respect to object detection, the number of vertical beams that can be expected to hit a given object is a function of the range to the object which limits the effective range for successful detections.

2.5 Summary

This chapter described some sensing technologies capable of generating 3D point clouds. Sensors specific to this dissertation were also described, the characteristics of which are summarized in tables 2.1 and 2.2. The information is split into two tables for readability. In the next chapter methods of calibrating these sensors for robotics applications are discussed.

	Carmine	LMS 291-S05	LMS 291-S14
Field of View	$57.5° \times 45°$	$180°$	$90°$
Angular Resolution	$0.09° \times 0.08°$	$1°$	$0.5°$
Points per Return	640×480	181	181
Scan Rate (Hz)	30	75	75
Dimensions (mm)	$250 \times 350 \times 1800$	$155 \times 156 \times 210$	$155 \times 156 \times 210$
Weight (g)	226	4500	4500

Table 2.1: Sensor summary (part 1). This table summarizes the characteristics of the Primesense Carmine, SICK LMS 291-S05, and SICK LMS 291-S14. The Field of View, Angular Resolution, and Points per Return rows for the Carmine are listed horizontal $\times$ vertical.

	O3D200	Actuated Hokuyo	VLP-16
Field of View	$40° \times 30°$	$270° \times 10°$	$360° \times 30°$
Angular Resolution	$0.625° \times 0.625°$	$0.25° \times \approx 0.8°$	$(0.1° - 0.4°) \times 2°$
Points per Return	64×48	≈ 8100	$(3600 - 900) \times 16$
Scan Rate (Hz)	7 (20 max)	≈ 5	$5 - 20$
Dimensions (mm)	$75 \times 95 \times 137$	$60 \times 70 \times 106$	103 (diameter) $\times 72$
Weight (g)	1250	≈ 530	830

Table 2.2: Sensor summary (part 2). This table summarizes the characteristics of the IFM O3D200, Actuated Hokuyo, and Velodyne VLP-16. The Field of View, Angular Resolution, and Points per Return rows are listed horizontal $\times$ vertical.

Chapter 3

Sensor Calibration

This chapter describes procedures for calibrating the sensors described in the previous chapter for effective use in robotics applications. There are two types of calibration procedures described here: intrinsic and extrinsic. Intrinsic calibration procedures are used to increase the accuracy of an individual sensor. For example, manufacturing variations can result in two sensors from the same manufacturer yielding slightly different measurements. Extrinsic calibration procedures are used to determine position and orientation of the sensor with respect to some coordinate frame of reference. The following sections describe an intrinsic calibration procedure to increase the effective range of Primesense based sensors, an extrinsic calibration procedure for 3D sensors, and an extrinsic calibration procedure for 2D scanning LIDARs.

3.1 Primesense Intrinsic Calibration

As mentioned in the previous chapter, the maximum advertised range of the Carmine 1.09 is 1.4 meters, but objects at depths farther than 1.4 meters could still be detected. The triangulation based nature of structured light sensors induces a nonlinear noise model of the form $|\delta z| \propto z^2|\delta d|$, where δz is the error in the depth observation, z is the actual depth, and δd is the error in disparity [21]. In other words, errors grow quadratically with depth. This can be mitigated by using an appropriate error model and adjusting the depth measurements accordingly. Unfortunately, global distortion models used for traditional camera calibration are of limited use as sensors based on the Primesense appear to have irregular distortion patterns unique to each individual sensor [22]. While they propose an unsupervised procedure to intrinsic calibration in

[22], we use an alternate approach that while supervised, is fast to use and significantly less complex to implement.

Starting at the minimum effective range of the sensor, the user captures a depth image of a nominally flat wall. The sensor is then moved incrementally farther from the wall, and a new image is captured out to the maximum sensor range. For example, if the minimum and maximum ranges of interest were 0.5 m and 3.0 m respectively, depth images would be captured at nominal depths of $\mathbf{z} = [0.5,1.0,1.5,2.0,2.5,3.0]$ meters. Note that the exact spacing is not critical. However, the accuracy of the depths $\mathbf{z}$ *is* the basis for the calibration, and must be measured accurately. This can be readily accomplished using standard tools (e.g., a tape measure or laser distance measurer). It is also important that the sensor's optical axis be roughly normal to the wall surface. To ensure this, we developed an application that provides visual feedback of the alignment error between the sensor's optical axis $\vec{o}$ and the wall's surface normal $\vec{n}_w$. This is estimated by using RANSAC [23] to automatically segment the wall plane in real-time. The user then adjusts the sensor orientation until $||\vec{o} \times \vec{n}_w|| \approx 0$. In practice, an alignment error of ≤ 1 degree is adequate for calibration, and easily obtained.

Given a set of k point cloud images $\mathbf{P} = [P_1, \ldots, P_k]$ and corresponding ground truth depth measurements $\mathbf{z}$, the remainder of the calibration process is completely automated. For each $P \in \mathbf{P}$, we recover the parameters for the respective wall planes $\Pi = [\Pi_1, \ldots, \Pi_k]$ where the relative orientation is again estimated using RANSAC and the translation using the depth measurements $\mathbf{z}$. Given robust estimates of the actual wall's relative positions and orientations Π, the point clouds $\mathbf{P}$ are adjusted to ensure that each point $p_i(i,j) \in P_i$ lies on its respective plane Π_i. This is accomplished by generating a set of scaling coefficients $\mathbf{K} = K_1, \ldots, K_k$ for each point of each point cloud. We denote the corrected point cloud set as $\mathbf{P}^*$.

The scaling coefficients $\mathbf{K}$ are to this point limited to the discrete set of ranges $\mathbf{z}$ where calibration data were collected. These are generalized to continuous space by modeling the scaling coefficients as a quadratic function of scene depth, i.e.,

$$K(i,j,z) = A(i,j)z^2 + B(i,j)z + C(i,j) \tag{3.1}$$

where (i,j) are the pixel coordinates of the point cloud. Thus, every sensor pixel has it's own specific quadratic function $k(i,j,z)$ that is used to determine the scaling factor at a given depth z. The quadratic coefficients $[A(i,j), B(i,j), C(i,j)]$ for each pixel (i,j) are recovered as a least squares solution minimizing the residuals between

$\mathbf{P}$ and $\mathbf{P}^*$. The coefficients are calculated offline, and stored in three Look Up Tables (LUTs) A, B, C corresponding to the respective quadratic coefficients.

A point cloud P of $m \times n$ points can be described through its Euclidean coordinates $X, Y, Z \in \mathbb{R}^{m \times n}$ where each matrix entry corresponds to the x, y, z coordinates of the respective point. To calculate the corrected points, the following operations are performed on the streaming point cloud:

$$
\begin{aligned}
K(i,j) &= A(i,j) * Z(i,j)^2 + B(i,j) * Z(i,j) + C(i,j) \quad \forall \ (i,j) \\
X^*(i,j) &= K(i,j) * X(i,j) \\
Y^*(i,j) &= K(i,j) * Y(i,j) \\
Z^*(i,j) &= K(i,j) * Z(i,j)
\end{aligned}
$$

where X^*, Y^*, Z^* denote the corrected point set. Thus, online intrinsic calibration can be performed at a cost of only several floating point operations and array look ups per point.

We have used the calibration procedure extensively, and performance has been very good. A sample calibration run is shown at Figure 3.1. The left sub-figure shows a point cloud before (top in red) and after (bottom in blue) calibration. Qualitatively, we see that both the distortion and dispersion of the points were significantly reduced. This is also reflected quantitatively in the center-right sub-figures, which show the mean error and mean standard deviation of the points vs. scene depth (pre-calibration and post-calibration). The reductions in both error and variance were significant, clearly demonstrating the efficacy of the approach.

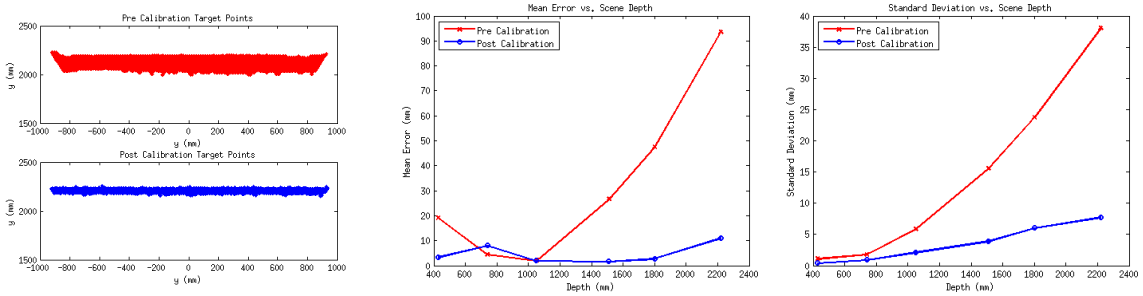

Figure 3.1: (Left) Sensor points before (top) and after (bottom) intrinsic calibration. Note that both point distortions and dispersion is reduced. This is also reflected in the mean error (center) and standard deviation (left) of the points.

3.2 3D Sensor Extrinsic Calibration Procedure

The goal of this procedure is to recover the rotation $\mathbf{R}_s^v$ and translation $\mathbf{t}_s^v$ relating the sensor frame $\mathbf{F}_s$ to the vehicle frame $\mathbf{F}_v$. To accomplish this, a calibration target is placed at constant x (depth) in $\mathbf{F}_v$ and with known y position. For the remainder of the procedure discussed, we assume a single 3D image $\mathbf{I}$ is taken of the scene, and used to perform the calibration procedure. In practice, a large number of images (*e.g.*, 10-100) could be taken, and an "average" calibration returned to mitigate random sensor noise. The entire process would take only several seconds.

The following assumptions are made for calibration purposes:

1. A vertical rectangular calibration target of known dimension is available.
2. The calibration target's position and orientation with respect to the vehicle frame is known.
3. The target orientation is normal with respect to both the ground plane and the x-axis of the vehicle frame.
4. The two largest planes in the sensor's field-of-view (FOV) are the ground plane and the target.
5. The orientation of the sensor with respect to the vehicle frame is poorly known (*i.e.*, within 45 degrees for a given axis.

To recover the sensor rotation, the two largest planes Π_1, Π_3 in the scene are recovered (the choice of the "3" subscript will become apparent shortly). This is accomplished by recovering Π_1 using RANSAC [23], removing its associated inliers from the point cloud, and then repeating the process with the reduced point set to obtain Π_3. This is illustrated in Figure 3.2, where the two segmented planes in the point cloud are highlighted in green and blue.

To identify which plane is the ground plane vs. the target, we compare the unit normals of the recovered planes $\mathbf{u_1}, \mathbf{u_3}$ in $\mathbf{F}_s$ with the associated normals in the vehicle frame, *i.e.*, $\mathbf{u_x} = [1, 0, 0]^T$ and $\mathbf{u_z} = [0, 0, 1]^T$. From Assumption 5, detailed above, we can correctly associate the planes in $\mathbf{F}_s$ with the planes in $\mathbf{F}_v$. Without loss of generality, assume that $\mathbf{u_1}$ is associated with $\mathbf{u_x}$. This yields two of the three basis vectors needed to infer the relative orientation of the sensor. The third is accomplished by taking the vector product $\mathbf{u_2} = \mathbf{u_1} \times \mathbf{u_3}$. Note that to ensure a proper right-hand

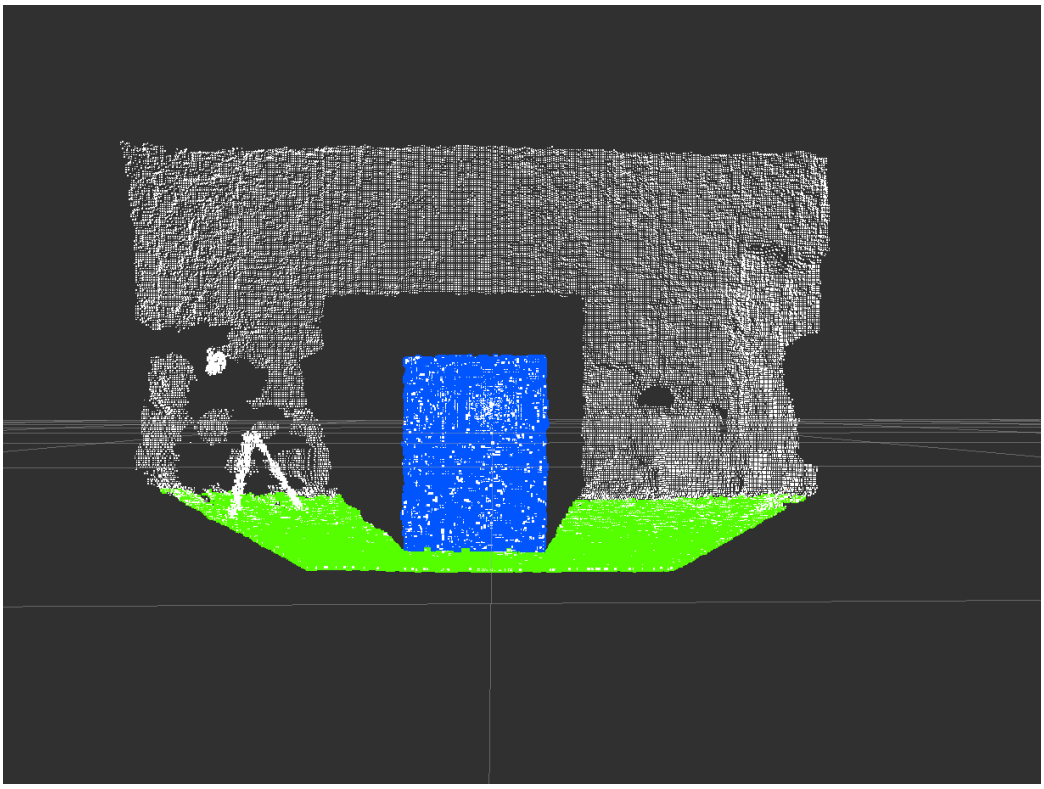

Figure 3.2: 3D calibration image with the ground plane (green) and calibration target (blue) segmented.

coordinate frame, the directions of the basis vectors $[\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}]$ can be validated by taking the dot product with the respective basis vectors in $\mathbf{F}_v$. If the projection is negative, then the sign of the normal is reversed, *e.g.*, $\mathbf{u_1} = -\mathbf{u_1}$.

The matrix $[\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}]$ is equivalent to $\mathbf{R}_v^s$, and $\mathbf{R}_s^v$ is readily obtained from its transpose. Note however that since the recovered planes Π_1, Π_3 will not be exactly orthogonal, the initial estimate for $\mathbf{R}_s^v$ will not be a valid rotation matrix. To remedy this, we use singular value decomposition (SVD) to refine $\mathbf{R}_s^v = UDV^T$ and obtain its closest valid rotation matrix by $\mathbf{R}_s^{*v} = UIV^T$ where I is the identity matrix.

After recovering $\mathbf{R}_s^{*v}$, all that remains is to recover the translation vector $\mathbf{t}_s^v = [t_x, t_y, t_z]^T$. The first component t_x can be obtained using the distance to the segmented target plane since by Assumption 2 its x position in $\mathbf{F}_v$ is known *a priori*. The same argument can be made for t_z using the segmented ground plane since we assume that in $\mathbf{F_v}$ for the ground plane $z = 0$. This leaves only t_y to be recovered.

Again, we know the y position of the target in $\mathbf{F_v}$. Also by Assumption 1, we

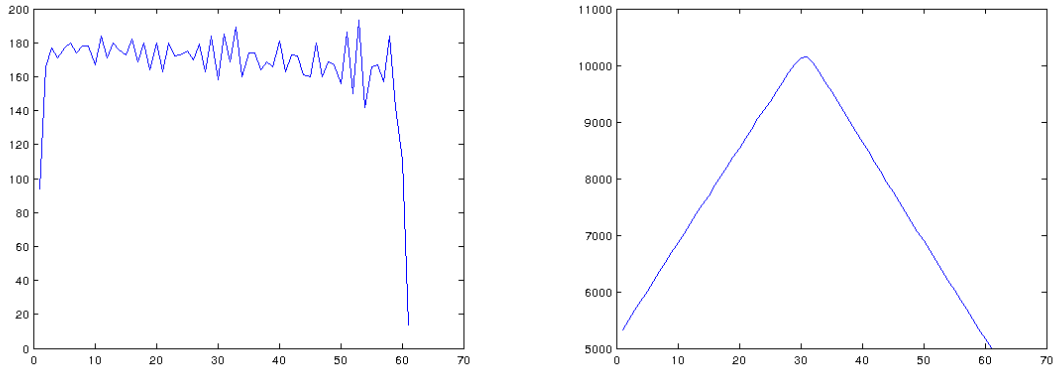

Figure 3.3: Target histogram $h(y)$ before (left) and after its convolution with a boxcar function. The peak of $f(y)$ corresponds to the target center y_c.

know its dimensions. We will exploit this knowledge to recover the y coordinate of the target center y_c. Since its position and size are known, this will allow us to estimate t_y.

To recover y_c, we first rotate all of the points in $\mathbf{F_s}$ to an intermediate coordinate frame using $\mathbf{R_s^{*v}}$. We then generate a histogram $h(y)$ by binning all of the target points along the y direction using a 1mm bin size. A representative result for $h(y)$ is shown at Figure 3.3 (left).

Next we convolve $h(y)$ with a discrete boxcar function $g(y)$ of unit amplitude and with the same interval width as the target discretized at 1mm resolution. The resulting function we denote $f(y) = g(y) * h(y)$. To illustrate with an example, the target used in this calibration example was 61 cm wide. Discretized to 1 mm, the resulting function $h(y) = [1, 1, \ldots, 1, 1] \in \mathbb{R}^{610}$ is a vector of 610 "1s". This convolution is equivalent to performing a crosscorrelation of the segmented target associated with $h(y)$ with a "perfect" target associated with $g(y)$. When they completely overlap, the value of $f(y)$ will be maximized. In other words, $y_c = \arg\max f(y)$. Again, this is illustrated in Figure 3.3, showing the target histogram $h(y)$ on the left and $f(y)$ on the right. As expected, latter is unimodal with a single, obvious maximum.

Figure 3.4 show sample results of a 3D image taken in the sensor frame $\mathbf{F_s}$, and then warped to the vehicle frame $\mathbf{F_v}$ using the recovered extrinsic parameters.

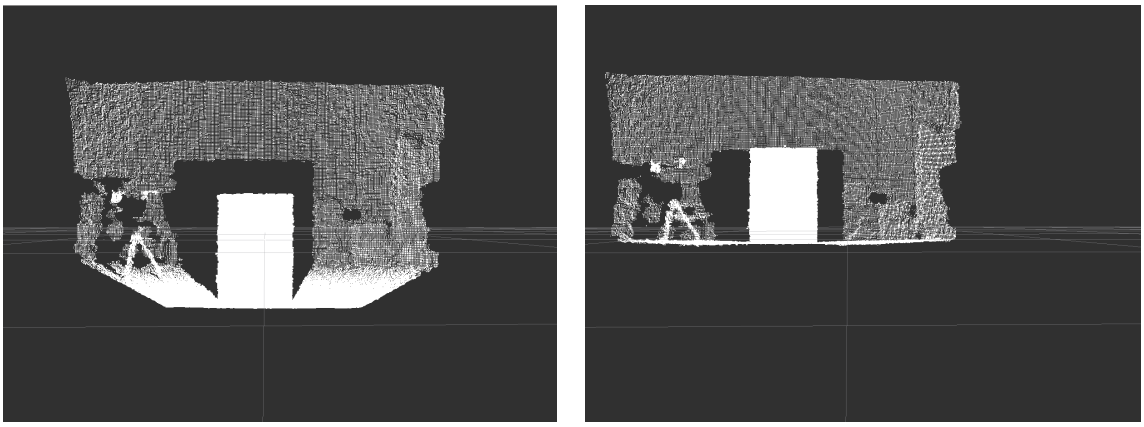

Figure 3.4: Original sensor image (left) and resulting image after being warped to the vehicle frame $\mathbf{F}_v$. Both the translation and rotation of points are apparent.

3.3 2D LIDAR Extrinsic Calibration Procedure

The procedure described here originally described in [24] and is summarized here for completeness. The goal of this procedure is to recover the rotation R_L^V and translation T_L^V relating the sensor frame L to the vehicle frame V under the assumption that the sensor returns a 2D scan. Because of this assumption, the points are registered to a common world frame W and we rely on the motion of the vehicle to recover the 3D extrinsic parameters. That is, a sensor point, registered to the world frame has the form

$$X_{i,W} = R_V^W(t)(R_L^V X_{i,L} + T_L^V) + T_V^W(t)$$

where the unknowns are R_L^V, T_L^V, and X_W and i is a corresponding point index, the parameters relating the vehicle frame to world frame, R_V^W and T_V^W, are functions of time. With a sufficient number of non-coplanar point correspondences between X_L, the point in the sensor frame, and X_W, the point in the world frame, we can solve for the R_L^V and T_L^V directly.

The following assumptions are made for this procedure:

1. The estimate of R_V^W and T_V^W is of high quality.

2. The sensor can return points with associated intensity values.

3. The distances between fiducial markers on the calibration targets are known.

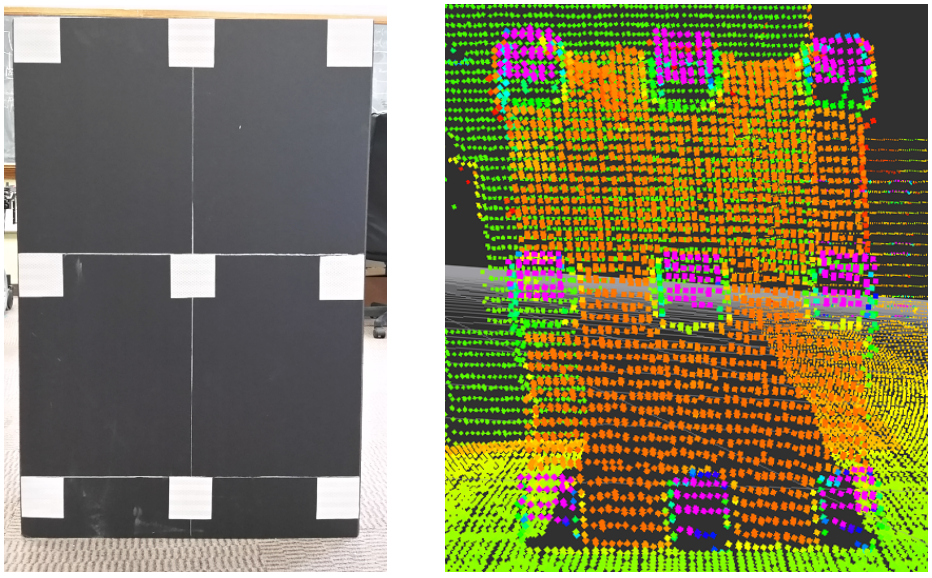

Figure 3.5: (Left) A photo of the calibration target. (Right) An aggregation of sequential LIDAR scans where the color corresponds to the remission values. The retro-reflective tape is clearly visible.

We obtain a set of points by setting up calibration targets in a space large enough to accommodate the vehicle and the sensor's range. An example of a calibration target (a poster board with with square pieces of retro-reflective) is shown in Fig. 3.5 (Left), that have retro-reflective targets to facilitate segmentation of feature correspondences. The laser scans are logged as the vehicle moves past the calibration targets. Afterwards, the laser scans are aggregated into a single point cloud and the targets are segmented using a threshold operation ($\gamma > \gamma_{min}$); an example of the point cloud is shown in Fig. 3.5 (Right). Note that targets are clearly not point features (each target is 5 cm $\times$ 5 cm). To reduce each target to point feature, the nearest data point to the centroid of each cluster was used. An actual data point was needed in order to retrieve the vehicle pose $(R_V^W(t), T_V^W(t))$ at which the measurement was taken.

To solve for the unknown R_L^V and T_L^V we use a similar procedure as described in [24]. We assume that we have an initial estimate $(\hat{R}_L^V, \hat{T}_L^V)$ obtained by physical measurements. We can linearize a small change in the Euler angles of a rotation matrix by starting with Rodrigues' rotation formula

$$R(\hat{n}, \theta) = \mathbf{I} + \sin\theta[\hat{n}]_\times + (1 - \cos\theta)[\hat{n}]_\times^2$$

where $\hat{n}$ is an axis, θ is a rotation about that axis, and$[\hat{n}]_\times$ denotes the matrix form

of the cross product operator with the vector $\hat{n} = [\hat{n}_x, \hat{n}_y, \hat{n}_z]^T$, that is

$$[\hat{n}]_\times = \begin{pmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{pmatrix}$$

and then applying the small-angle approximation, which yields

$$R(\omega) \approx \mathbf{I} + \sin\theta[\hat{n}]_\times \approx \mathbf{I} + [\theta\hat{n}]_\times = \begin{pmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{pmatrix}$$

where $\omega = \theta\hat{n} = [\omega_x, \omega_y, \omega_z]^T$. We can then model R_L^V as the product initial estimate and a small change in the Euler angles, that is, $R_L^V \approx \hat{R}_L^V R(\omega)$.

We then cast the problem of solving for (R_L^V, T_L^V) as a second-order cone program which allows us to enforce known geometric constraints. If we assume that our initial estimate $\hat{R}_L^V$ is accurate within a small tolerance on the Euler angles, we can write a second-order cone constraint of the form $\|\omega\|_2 \leq \delta$ for each of the three Euler angles. A bound on errors in T_L^V could be written as $\|T_L^V - \hat{T}_L^V\|_2 \leq \epsilon$. Furthermore, we know the approximate height of each data point since we are collecting data indoors. This knowledge can be encoded as a second-order cone constraint of the form $\|X_{i,W}^* - T_i\|_2 \leq \beta_i\|_2$ where $T_i = [0, 0, b]^T$ and b is the known height of the target i's centroid. The optimization program is written as

$$\begin{aligned} \min_{\Delta R_L^V, T_L^V, \hat{X}_{i,W}} \quad & t \\ & \sum_{i=1}^{N} \|\hat{X}_{i,W} - X_{i,W}^*\|_2^2 \leq t \\ & \|\omega_x\|_2 \leq \delta_x, i = 1 \ldots N \\ & \|\omega_y\|_2 \leq \delta_y, i = 1 \ldots N \\ & \|\omega_z\|_2 \leq \delta_z, i = 1 \ldots N \\ & \|T_L^V - \hat{T}_L^V\|_2 \leq \epsilon_i, i = 1 \ldots N \\ & \|X_{i,W}^* - T_i\|_2 \leq \beta_i, i = 1 \ldots N \end{aligned}$$

However, due to the linearization of the original problem, the recovered rotation matrix is not necessarily in the group $SO(3)$ or the globally optimal solution. To properly orthogonalize R_L^V we perform a singular value decomposition $R_L^V = UDV^T$ and retrieve a valid rotation matrix $R_L^{*V} = UV^T$. But, this orthogonalization invalidates the local optimality of the solution. To mitigate this effect, the procedure was performed a second time using R_L^{*V} as the initial estimate.

Chapter 4

Approach to Robot Navigation

As mentioned in Section 1.3, the main capabilities of a robot navigation system are: map construction and interpretation, localization, and path planning. While a major focus of this dissertation is the 3D point cloud based perception component, we also want to demonstrate autonomous navigation systems. This requires the integration of the 3D perception component with the navigation systems. This chapter describes our general approach to each of the navigation system's components.

4.1 Map Representation

The internal representation of the environment used in our robotic navigation applications is a feature-based map, where each feature in the map denotes a landmark of semantic interest. Associated with the landmark map is a route network of waypoints for the purpose of path planning. Additionally, an instantaneous occupancy map representation is used for local path planning and dynamic obstacle avoidance. This section describes the global map representation components: the landmark map and the route network. The local occupancy map is described below in Section 4.3.1.

4.1.1 Landmark Map

A landmark map is a feature-based map representation where each feature in the map is a landmark that is perceptually differentiable from its surroundings. The representation for features in our work are the 2D coordinates with respect to some global frame of reference and some additional semantic information to assist in data

association for the localization procedure. The assumption here is that the environments that the wheelchair system traverses are approximately planar, so using a 2D map representation reduces the computational complexity for the localization and path planning systems. This choice does not necessarily limit 3D based landmark detection as the geometry of the detected landmarks can be projected to the 2D map coordinate frame.

4.1.2 Route Network

The route network is modeled as a graph $G(V, E)$ where waypoints correspond to the vertices $v_i \in V$ and the edges $e_{ij} \in E$ correspond to directed edges connecting pairs of waypoints. Each edge can also contain semantic information associated with it, for example a speed limit, which could be used to constrain the sampled trajectories of the local path planner 4.3.3. The purpose of the route network is to demarcate a safe path with respect to the landmark map.

4.2 Localization

The localization method used in this dissertation is a Rao-Blackwellized particle filter approach similar to the FastSLAM 2.0 algorithm [25]. The FastSLAM algorithm is a solution to the simultaneous localization and mapping problem. Since our landmark map was known *a priori*, we were only concerned with the localization aspect. The particle filter approach was chosen because in our applications it was more robust to aspects of the environment that were not modeled by the system than an EKF based localization approach.

A set of p particles was maintained where each particle had the form:

$$Y_k^{[p]} = \langle x_k^{[p]}, \langle \mu_1^{[p]}, \Sigma_1^{[p]} \rangle, \ldots, \langle \mu_N^{[p]}, \Sigma_N^{[p]} \rangle \rangle, \tag{4.1}$$

where $x_k^{[p]}$ was the pose of the p^{th} particle at time k defined as $[x, y, \theta]^T$ where x and y were the Cartesian coordinates and θ was the robot's orientation with respect to the map's frame. Every particle was initialized with a map containing the mean, $\mu^{[p]}$, and covariance estimate, $\Sigma^{[p]}$, for each landmark in the map. The mean vector had the form $[x_l, y_l, s]^T$, where x_l and y_l were the landmark's position and s was the landmark's signature, which contained additional semantic information to facilitate data association.

The prediction phase of the filter consisted of sampling from a probabilistic motion model of a differential drive robot where the control inputs (v, ω) were corrupted with additive Gaussian noise [26]; two noise parameters were used for error in translational movement due to linear and rotational motion (a_1) and $a_2)$ as well as two for the error in rotational movement due to linear and rotational motion (a_3) and (a_4).

Data association of segmented landmarks was done by using a maximum likelihood correspondence. During the correction phase of the filter, for a given particle p, every observation at time k was compared to each of the landmarks in p's map and a weight was computed for each observation - landmark pair that measured the likelihood that the observation corresponded to the landmark. Observations were of the form: $z = [\rho, \phi, s]^T$, where the components corresponded to the range, bearing, and signature (radius of the pole). The weight was approximated by a Gaussian with mean $(z - \hat{z}_j)$, where z is the measured observation and $\hat{z}_j$ is the predicted observation of particle p's j^{th} landmark, and covariance matrix $Q_j = H_j \Sigma_j^{[p]} H_j^T + Q$, where H_j is the Jacobian taken with respect to the map features, $\Sigma_j^{[p]}$ is the covariance of particle p's j^{th} landmark, and Q is the linearized measurement noise. The weight assigned to each association was the Mahalanobis distance, defined as:

$$w_j = |2\pi Q_j|^{-1/2} \exp\left(-\frac{1}{2}(z - \hat{z}_j)^T Q_j^{-1}(z - \hat{z}_j)\right). \tag{4.2}$$

The landmark with the maximum weight was chosen to be associated with an observation if it exceeded a minimum threshold. We found that the addition of the signature led to fewer false data associations than using location alone.

4.3 Path Planning

The path planning method is composed of two main tasks: mapping the immediate local environment from sensor data and planning a path through this local environment.

4.3.1 Constructing the Local Map

Our robot application all employ a local map modeled as a 2D occupancy grid for the purpose of generating local plans. This local map was centered at the position of the robot and moved with the robot in a rolling window fashion. We leveraged ROS [27] for populating and clearing cells in the local map via raytracing techniques.

For navigation purposes, 3D points from any equipped sensors were projected down to the 2D occupancy grid M where each cell in the grid that contained a projected point was classified as occupied. Each occupied cell $M(x, y)$ was given an obstacle cost value $C_{obst}(x, y) = \infty$. Nearby cells were also assigned cost values based on the proximity to occupied cells as well as the footprint of the robot; if the robot were to occupy a cell $M(x, y)$ and any portion of its footprint would overlap an obstacle cell where $C_{obst}(x, y) = \infty$, then that cell was also assigned a value of ∞ making it untraversable by the local planner. Otherwise, obstacles were modeled by exponential potential functions.

In addition to the occupancy cost for each cell, the local map also maintained planning related costs for each cell $C_{goal}(x, y)$ and $C_{path}(x, y)$. C_{goal} was proportional to the distance from the current robot position and the subgoal position G_i (the closest waypoint in the route network to the goal waypoint that is within the bounds of the local map M) which was given a zero cost value. Similarly, C_{path} was proportional to the distance from the robot's position to the cells along the waypoint path where path cells were determined by linear interpolation between the waypoints and assigned a cost of zero. The resulting occupancy map M and associated costs C_{obst}, C_{goal}, and C_{path} were used by the local planner for trajectory planning. Fig. 4.1 depicts an example of a local map.

4.3.2 Global Path Planning

The global planner for our applications operates on the route network map representation, that is, given the waypoint network, $G(V, E)$, described in Section 4.1.2 and a desired destination $v \in V$, the path to the destination is computed via Dijkstra's algorithm. Note that the weight for each edge does not necessarily have to be distance and could be a function of any semantic information contained in the edge so the path returned from Dijkstra's algorithm could minimize some other quantity. For example, an edge may contain a semantic value corresponding to some quantification of safety, and the weights could be a function of safety and distance, and the shortest safest path is returned.

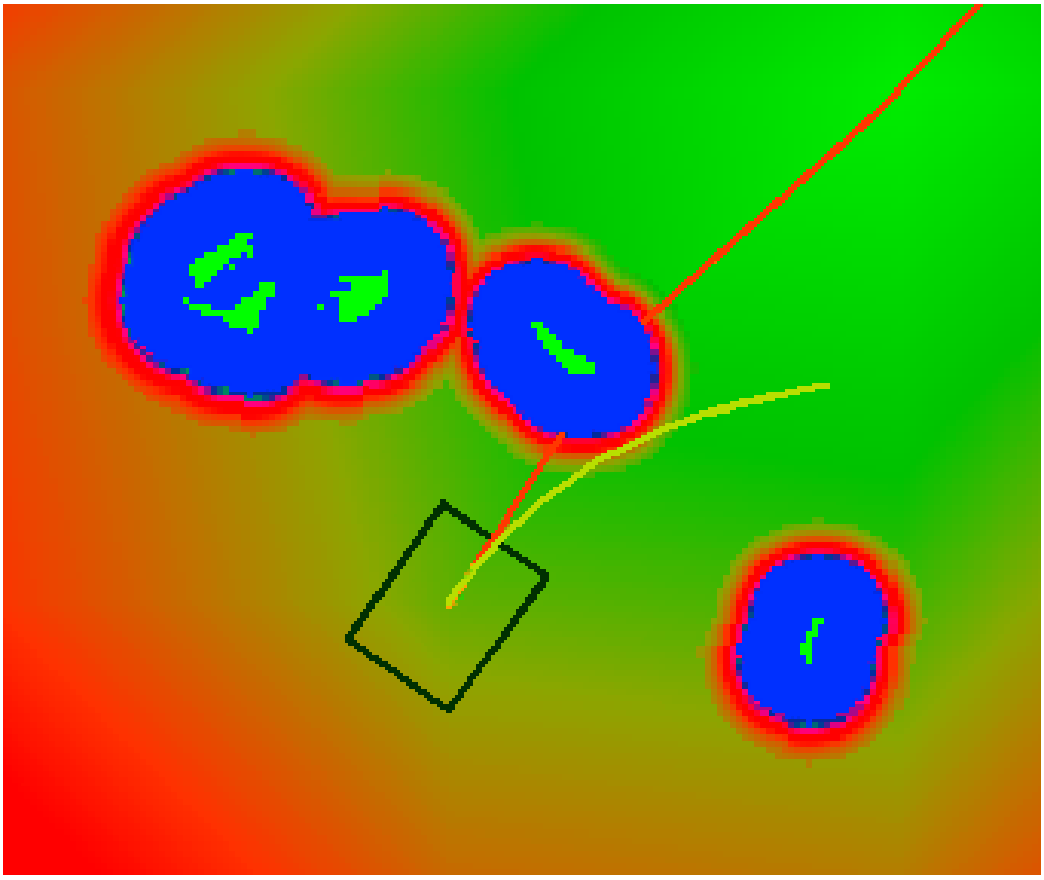

Figure 4.1: Navigation visualization. The black rectangle is the robot footprint, the red line is the desired path, the yellow line is the lowest cost trajectory. The bright green cells are obstacles of maximum cost. Obstacle cells are inflated with a high cost region in blue.

4.3.3 Local Path Planning

A sample based approach was used for local planning where the input space ranges over linear and angular velocities (v, ω) [28]. Sampling control velocities in this way ensured that the trajectory honored the kinematics of the SWS. At the beginning of each planning cycle, a set of trajectories of the form:

$$T_i = (\mathbf{x}, v_1, \omega_1, \ldots, v_n, \omega_n), \tag{4.3}$$

were sampled over the range of velocities $v \in [0.1, 1.2]$ m/s and $\omega \in [-0.3, 0.3]$ rad/s where $\mathbf{x}$ is the pose of the SWS and n denotes the number of discrete time steps in the control horizon. Each sampled trajectory T_i was then evaluated with a cost function of the form:

$$C(T_i, M) = C_{obst} + k_1 C_{goal} + k_2 C_{path}, \tag{4.4}$$

where M is the occupancy map described in Section 4.3.1 and k_1 and k_2 are tunable gain parameters. C_{obst} was the maximum obstacle cost of any cell along the specified trajectory. If $C_{obst} = \infty$, the trajectory was infeasible as it passed through an obstacle and was discarded. The goal and path costs were determined by the endpoint of the trajectory (x', y') and assigned the values $C_{goal}(x', y')$ and $C_{path}(x', y')$. The optimal trajectory $T^* = \text{argmin } C(T, M)$ was selected and the associated velocity command $(v^*, \omega^*) \in T^*$ was issued to the robot's Motion Control Module (MCM).

Chapter 5

CoPilot: Autonomous Doorway Navigation

This chapter describes CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an electric powered wheelchair (EPW) for automated doorway detection and traversal. The goal of CoPilot is to leverage 3D sensing to provide reliable doorway traversal at the level of a typical human operator or better. The following sections detail the motivation for the CoPilot system, some related work, a description of the development platform, the details of doorway detection and doorway navigation approaches, experimental results demonstrating the reliability of the system, and a brief discussion.

5.1 Background

The U.S. Department of Health and Human Services reports that the number of people over the age of 65 will increase from 40.4 million people in 2010 to over 70 million by 2030 [29]. This rapid growth in the U.S. elder population will also increase the number of people with age-related symptoms that hamper their mobility. Such common symptoms include visual impairments, dementia, and Alzheimer's disease [30]. Providing electric-powered wheelchairs (EPWs) to seniors (and others) is a significant step in helping them live at home and maintain independent mobility. However, it is not without its own challenges. Maintaining straight paths and avoiding obstacles is often challenging - especially for drivers using alternative controls such as sip-and-puff devices, switch driving systems, chin controls, or short-throw joysticks.

Additionally, traditional joystick users with impaired hand control and those who rely on "latched driving" modes (i.e., cruise control) for independence and function may require additional assistance to ensure safe and comfortable mobility. To realize the home health benefits of EPWs while also maintaining safety, active safety systems for EPWs could be deployed.

To this end, we have developed CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an EPW. Similar to active driver-assist systems in automobiles, the driver remains in primary control of the vehicle, while in the background, CoPilot uses intelligent sensing and drive control systems that work in cooperation with the driver to aid in avoiding obstacles/collisions and fine precision driving tasks. The motivation is that as an individual begins to lose cognitive, perceptive, or motor function due to age, injury, or disease, CoPilot can augment that loss because it can interpret the user's intent by seeing into the environment. This exteroceptive sensing capability is enabled by leveraging the latest in three-dimensional (3D) imaging technology. While being developed with a suite of semi-autonomous driving behaviors in mind, the focus of this system is automated doorway detection and traversal. This functionality was motivated by discussions with physical and occupational therapists in the wheelchair space who prioritized doorway navigation as a capability that would provide real value to EPW users. CoPilot provides near 100% effectiveness in this application.

5.2 Related Work

Doorway detection using 3D sensing has been accomplished in various ways. Rusu et al. used 3D point clouds to locate doors [31]. The goal was to find doors for the purpose of opening or closing them with a robotic manipulator. When the robot was at a door location, a planar model was fit to the point cloud data. The models were validated based on geometric constraints. More recently, RGB-D data has been used for the task of parsing indoor scenes [32, 33]. The goal of which is to detect and correctly label objects in indoor environments. This is a more difficult task than looking for a single category of object, in our case doorways. These algorithms are based on learning classifiers where the feature vectors are largely inspired from computer vision techniques, such as histograms of oriented features. In our work, we also leverage computer vision approaches for some aspects of doorway detection.

Early approaches of wheelchair systems capable of doorway traversal include [34–36]. For navigation, Levine et al. [34] and Yanco [35] both utilized an array of sonar sensors and Parikh et al. [36] used a planar laser scanner. While these works yielded successful demonstrations, the limitations of the sensors were not necessarily suitable for use in cluttered environments. For example, depending on sensor placement, these approaches might be susceptible to navigating through a table because the table legs could be detected but not the table top.

The work most similar to our own is Derry and Argall [37], where the goal was to detect open doorways suitable for wheelchair traversal. Their approach involved processing point cloud data to fit planar models under the assumption that gaps in the planar model correspond to doorways if they meet certain geometric criteria. A key difference in approaches is that while their focus was in processing point clouds, our algorithms emphasize processing the depth images directly. Furthermore, their investigation was limited to doorway detection. In contrast, CoPilot provides a complete solution for automated doorway navigation.

5.3 Development Platform

The development platform used in this research was based on the Quantum Q6 Edge electric powered wheelchair (EPW) shown in Fig. 5.1. The Q6 features motors with integrated encoders for measuring wheel velocities. To access these for odometry purposes, we interfaced an on-board embedded computer with the EPW's motor controller over the CAN bus. It also enabled the regulation of the EPW's linear and angular velocities via a software-based PID.

Exteroceptive sensing was from two Primesense Carmine 1.09 sensors. The Carmine 1.09 is the shorter range version of the Primesense structured lighting sensor. It has an advertised effective range between 35-140 cm (compared to 80-350 cm for the standard range Carmine 1.08). The decision to use the short range variant was to ensure that doorways and obstacles remained visible in close proximity to the chair. However, the maximum range of 1.4 m was extremely limiting. We addressed this through an intrinsic calibration procedure which extended the effective range to approximately 3 meters with little degradation in accuracy. This is discussed in detail in Section 3.1. Two sensors were used in order to increase the total field of view.

Figure 5.1: CoPilot integrated into an Quantum Q6 Edge EPW. The Primesense Carmine 1.09 sensors are circled in red.

This ensured better coverage of the chair footprint (to avoid collisions with obstacles), as well as facilitated doorway detection at a range of chair orientations. The mounting positions of the sensors are depicted in Fig. 5.1. Note that the sensors are mounted vertically rather than horizontally as this was found to be a less obstructive configuration.

5.4 CoPilot Perception

This section describes the components of the doorway detection procedure used by the CoPilot system.

5.4.1 Depth Image Warping & Fusion

Our approach to doorway segmentation relies heavily upon the observation that doorway border features are strongly vertical. We further observe that computationally, these features can be extracted most efficiently if the sensor frame is aligned vertically with the world frame, i.e., the gravity vector. An analogy would be the motivation for rectification of stereo image pairs. As a result, we warped and fused the depth image pair as a pre-processing stage.

Given two point clouds P_L, P_R associated with the left and right sensors, respectively, the first step was to warp the points to a common coordinate frame $\mathbf{F}$. We chose $\mathbf{F}$ to be centered between the actual sensor positions, and with an orientation identical to the EPW vehicle frame. Using the extrinsic calibration relating the sensor and vehicle frames, we recovered the rigid transformation between the frames and transformed the points in each point cloud

$$\hat{P}_L = {}^{C}R_L P_L + {}^{C}\mathbf{t}_L \tag{5.1}$$

$$\hat{P}_R = {}^{C}R_L P_R + {}^{C}\mathbf{t}_R \tag{5.2}$$

where $({}^{C}R_L, {}^{C}\mathbf{t}_L)$ and $({}^{C}R_R, {}^{C}\mathbf{t}_R)$ were the rigid transformations relating the left and right sensor frames to $\mathbf{F}$. Since most of our processing will be in the depth image space, we next calculated the back projection of $\hat{P}_L, \hat{P}_R$ to form the fused depth image I_D. In doing so, a couple of subtleties needed to be addressed. First, the back projection of points do not lie on exact pixel boundaries. As a result, we use a nearest neighbor interpolation scheme to form the depth image. Second, there was the potential that a point in both $\hat{P}_L$ and $\hat{P}_R$ would warp to the same pixel $I_D(i, j)$. In this event, the depth of the closer point was used.

The process is reflected in Figure 5.2. The left-center sub-figures show the raw depth images from the left and right sensors. Note that when mounted on the EPW, the sensors were rolled approximately 90 degrees which explains the vertical orientation of the depth images. The right sub-figure shows the resulting depth image I_D after transforming and fusing the point clouds. All subsequent image and point cloud processing is done using this image as input.

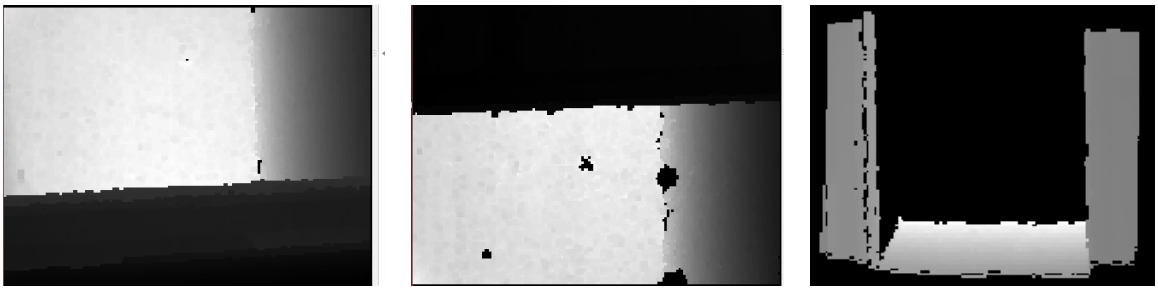

Figure 5.2: (Left-Center) Raw depth images of a doorway from the left and right sensors. (Right) Fused depth image.

5.4.2 Real-time Doorway Detection

After the transformation outlined in Section (5.4.1), vertical edges in the real-world map to vertical columns in I_D. The doorway detection procedure exploits this fact to efficiently find doorway boundaries based on salient features in the depth image. We evaluated two approaches to finding doorway boundaries, a feature based approach and a histogram based approach. After a set of doorway boundaries was obtained (from either approach), they were then validated based upon geometric constraints. We now describe the process in detail.

Feature Based Doorway Boundary Detection

Doorways are transition features between interior and exterior space. When viewed within a depth image I_D, they appear as spatial discontinuities. This is to be expected, as there must be sufficient free space to accommodate pedestrian (or EPW) traffic across the spaces. We leveraged techniques traditionally used in 2D image processing to localize this discontinuity, and by association the doorway edges. To enhance these edges, we convolved I_D with a $[-1, 0, 1]$ kernel to generate the horizontal gradient image, and then thresholded based upon the size of the depth discontinuity to generate an edge image E_D. The next step was to identify edges of sufficient length to be classified as a doorway edge. Note that simply summing the edge pixels for each column of E_D would produce incorrect results for two reasons: (i) the edges could actually be at different depths in 3-space, possibly corresponding to multiple objects, and (ii) the resulting sum would be biased towards objects close to the sensor because they subtend more pixels.

The first problem was mitigated by calculating the median depth $\bar{z}_k$ of each column k of I_D and generating a copy of the depth image, M_D, where values in column k are set to zero if they are not within some specified distance to $\bar{z}_k$. The idea was that true doorway edges would represent the majority of the edge length in the column, and the median value would therefore lie upon this edge. The second problem was addressed by weighting the depth measurements with the height of the unit pixel p_h subtended at the respective depth. The approach can be expressed concisely as

$$\Phi = \mathbf{1}^T (p_h \cdot E_D \odot M_D) \tag{5.3}$$

where $\mathbf{1}$ is a column vector of all ones, $\odot$ denotes elementwise multiplication, and Φ defines a row vector where each component corresponds to the edge height in each column. Each component in Φ was evaluated based on a minimum height requirement. The set of columns that meet the threshold were marked as potential doorway boundaries at a depth of $\bar{z}_k$.

The process is illustrated in Figure 5.3. The left sub-figure shows the edge image E_D. The center image shows edge pixels overlaid on the fused RGB-D image. The right image shows edge clusters projected to the $x - y$ plane. Note that each cell represents a potential doorway boundary, so that multiple candidates can be obtained from a single doorway image. Discriminating the correct edge (e.g., the front doorway edge vs. the rear) will be discussed in Section 5.4.2.

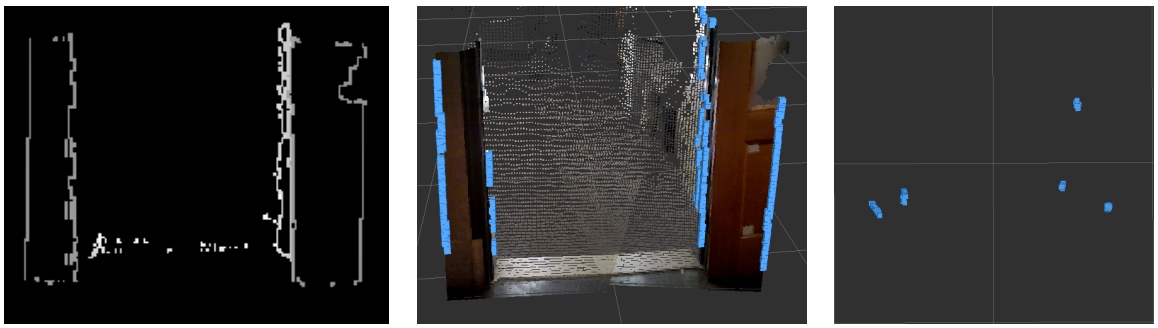

Figure 5.3: (Left) Edge image of doorway. (Center) Edge pixels identified in the scene. (Right) Top down view of door edge coordinates.

We quickly determined that by themselves, doorway edges were an insufficient feature for doorway detection. For example, an inward opening door may not offer a strong edge on the hinge side as the door face can provide a smooth transition into

the room. As a result, we also integrated corner features into our classifier. To do this, we first generated a 2D histogram $H(x, y)$ that bins points in 3-space to the ground plane. After applying the Harris operator to $H(x, y)$ [38], we identified the set of bins $\mathbf{C}$ in $H(x, y)$ that corresponded to corner features using an appropriate threshold. Marking a column as a potential doorway based on $\mathbf{C}$ required a small amount of effort since measurements from multiple columns could fall into the same bin. For each $C_k \in \mathbf{C}$, we found the data point $\mathbf{x}$ closest to the centroid of the bin and marked the associated column as a potential doorway boundary at a depth equal to the distance to $\mathbf{x}$.

The corner detection process is illustrated in Figure 5.4. The left sub-figure shows the fused depth image. The center image shows corner pixels overlaid on the fused RGB-D image. The right image shows valid corners projected to the $x - y$ plane.

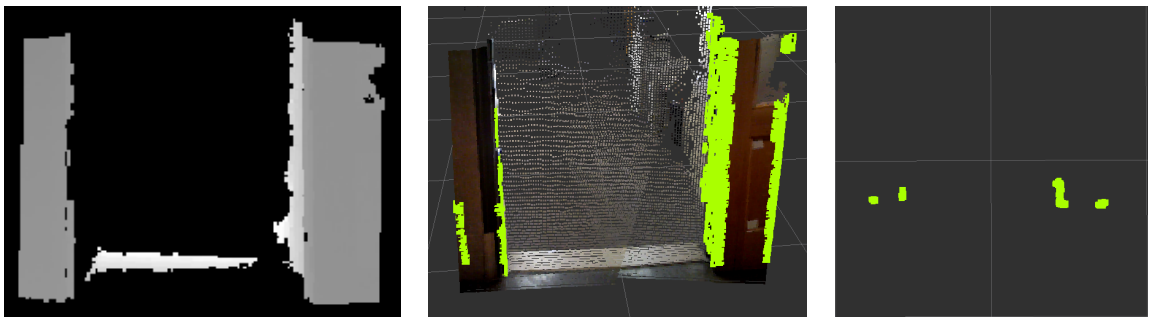

Figure 5.4: (Left) Fused depth image of doorway. (Center) Corner pixels identified in the scene.(Right) Top down view of doorway corner coordinates.

In practice, our feature based approach was very successful at segmenting doorways. However, its computational complexity - dominated by the corner segmentation component - was of concern. This motivated our investigation into the histogram-based approach described below.

Occupancy Histogram Doorway Boundary Detection

Our feature-based approach attempts to directly identify the doorway boundaries, leaving only a small number of candidates as input to the validation procedure outlined in Section 5.4.2. However, this comes at the expense of significant up-front computation. As a result, we investigated a simpler descriptor. It is based upon the observation that the segmented edge and corner features were subsets of all columns

largely occupied by a vertical object. For edge and corner features, we expend significant computational resources verifying that neighboring columns in 3-space are not occupied. But what if we simply identified each column that had a high occupancy rate as a potential doorway boundary? Undoubtedly this would lead to a much larger number of candidates for validation, but in practice the computational savings in image and point cloud processing more than makes up for this expense.

In effect, the depth image was reduced to a 1-D occupancy histogram. To accomplish this, we simplified the approach summarized in (5.3) to

$$\Phi = \mathbf{1}^T (p_h \cdot M_D) \tag{5.4}$$

which yielded a row vector where each component was the height of the object in each column corresponding to the median value $\bar{z}_k$. In other words, where in (5.3) we accumulated edge lengths, in (5.4) we are accumulating object height. Φ is now a 1D histogram of heights per bin where each bin corresponds to a column in M_D. Thresholding each component of Φ on a minimum height requirement segments every column that corresponds to a large vertical object.

When combined with the validation procedure in Section 5.4.2, this approach worked surprisingly well in practice. Compared to the feature-based approach, the implementation is far simpler as neither edge nor corner detection is required. It is also more efficient computationally. With a Primesense at VGA resolution (640×480), the feature based approach detected doorways at 12 Hz on the computer in Section 5.3. By comparison, the histogram approach ran at frame-rate (30 Hz). In the current version of CoPilot, the occupancy histogram approach is used exclusively.

Doorway Validation

Given the columns marked as candidate doorway boundaries and the associated depth depth values, the role of the doorway validation procedure is to find the best estimate of the relative position and orientation of the doorway. Algorithm 1 VALIDATE-DOORS outlines the procedure of reducing the set of doorway boundaries to a set of doorway candidates $\mathbf{D}$. Each pair of doorway boundaries must meet geometric constraints based on the width of the doorway (line 5), the orientation of the EPW to the doorway (line 5) and the amount of free space beyond the sill of the doorway (lines 10-14). Guided by the American's with Disability Act (ADA) [39] accessibility guidelines, minimum and maximum doorway widths were set to 82 cm and 162 cm,

respectively. The orientation constraint was set to $\pm 45^{\circ}$ and the free space beyond the doorway had to be sufficient to accommodate the EPW footprint. Doorway width and orientation validation are performed by the GEOMETRIC-VALIDATION sub-procedure.

Computationally, the most expensive part of VALIDATE-DOORS is the INTERSECT sub-procedure which verifies that sufficient free space exists beyond the candidate doorway via ray-tracing. In theory, there could be $O(n^2k)$ calls, where n is the number of columns and k is the number of free space tracing operations per doorway boundary pair. In practice, this will not happen due to constraints on doorway width, sensor field-of-view, and wheelchair orientation. When benchmarked with a single Primesense at VGA resolution, VALIDATE-DOORS had a mean run time of approximately 3 ms with a standard deviation of approximately 1 ms.

Algorithm 1 Door Validation

```
1: procedure VALIDATE-DOORS(O, B)   ▷ O: obstacle coordinates, B : boundary coordinates
2:     D ← ∅                                                  ▷ set of valid doorways
3:     for i ← 0 to n − 1 do
4:         for j ← i + 1 to n do
5:             if GEOMETRIC-VALIDATION(B[i], B[j]) then
6:                 continue
7:             end if
8:             is_valid ← true
9:             for k ← i to j do                                      ▷ trace free space
10:                p ← INTERSECT(B[i], B[j], k)        ▷ line segment intersection point
11:                if ‖O[k]‖ < ‖p‖ then
12:                    is_valid ← false
13:                end if
14:            end for
15:            if is_valid = true then
16:                D ∪ {[x, y, θ]^T}                                ▷ add doorway pose
17:            end if
18:        end for
19:    end for
20: end procedure
21: Note: The loops on B continue early when the column has no associated doorway boundary.
```

The doorway validation procedure returns the set of valid doorways **D** with the relative position of the doorway's center and its orientation. Note there is high probability that the classifier will return multiple doorway candidates. However, these

will typically be variants of the actual doorway opening (e.g., front edge to rear edge, front corner to rear edge, door stop to front corner, etc.). To ensure consistent position and orientation estimates, we wish to identify only the front edges/corners of the doorway. To this end, we use a heuristic of choosing the closest doorway candidate. In practice, this has worked quite well for detecting the actual doorway.

The process is illustrated in Figure 5.5. The center sub-figure shows the valid doorway candidates (red arrows), and the right sub-figure the chosen doorway. The latter well approximates the doorway position and orientation.

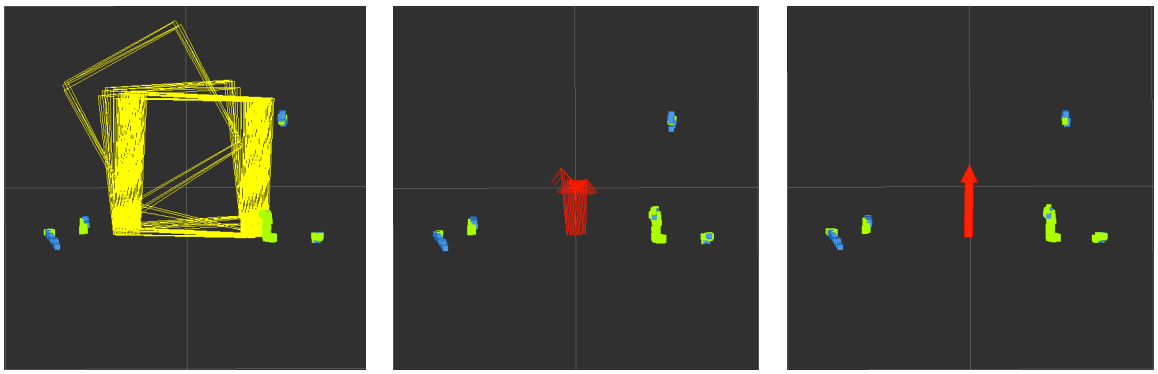

Figure 5.5: (Left) Free space check for feature pairs. (Center) The set of valid doorway features. (Right) The final doorway chosen using the "nearest doorway" doorway heuristic.

5.5 Autonomous Doorway Navigation

At the user level, the CoPilot interface is very intuitive. The user switches the EPW controller drive mode to "CoPilot" and manually drives towards the door. As soon as CoPilot detects the doorway, an icon appears on the LCD control panel. The user then pushes a single button to effect doorway traversal. Note also that the user can also steal back control from CoPilot at any time by simply touching the joystick.

At the software level, doorway navigation is decomposed into two primary subtasks: mapping the environment, and given such a map perform real-time planning and control of the EPW for safe and reliable doorway traversal.

The global planner for doorway navigation is very intuitive. Given a doorway position and orientation, the it constructs an objective path down the doorway centerline with the same orientation as the doorway itself. A goal pose $G = [x_g, y_g, \theta_g]$

is then placed on this path the length of the EPW through the doorway. This is a basic route network (§ 4.1.2) containing only two nodes.

For local planning, the sample based approach on the input space of the linear and angular control velocities (v, ω) described in Section 4.3.3 was used. The range of velocities sampled for this application was $v \in [0.1, 0.4]$ m/s, and $\omega \in [-0.3, 0.3]$ rad/s. Updates to the costmap were performed asynchronously whenever a scan from either of the sensors was available, with an objective feedback rate of 15 Hz. The costmap was centered on the EPW and covered an area of 10×10 meters with a grid cell resolution of 5 cm.

5.6 Experiments

The doorway navigation behavior for CoPilot is extremely effective. ADA compliant doorways can be navigated with near 100% reliability. The mapping capability also allows CoPilot to identify both static and dynamic obstacles in the environment, and react to these accordingly (i.e., by avoiding the obstacle or stopping when necessary). As additional anecdotal evidence of its performance, CoPilot was demonstrated at the headquarters of a major EPW manufacturer. The system was fully integrated into an EPW with a user-friendly interface. When placed in CoPilot driving mode, an icon would appear on the EPW's control display when a doorway was detected. The user then simply pressed a button to initiate door traversal. Although no data was collected during the demonstration, the system was tested by numerous company representatives across a large population of doors. CoPilot successfully traversed every door that the participants attempted.

To support this research, a more formal experiment was conducted over the course of several days at various locations around the Lehigh University campus. During this time, the EPW was operated in a natural fashion with no attempt to specifically align the wheelchair into a favorable pose. A total of 100 traversals of 100 unique doorway instances were attempted. All were successful. Fig. 5.6 depicts a sample of the doorways that were traversed. Note that CoPilot was even successful navigating through doorways where structured lighting systems might be expected to struggle, e.g., doorways with glass doors. Fig. 5.7 shows the variety of starting EPW poses and a probability mass function of the door widths. Note also that the large majority of the doorways were at the lower range of ADA compliant doorway widths.

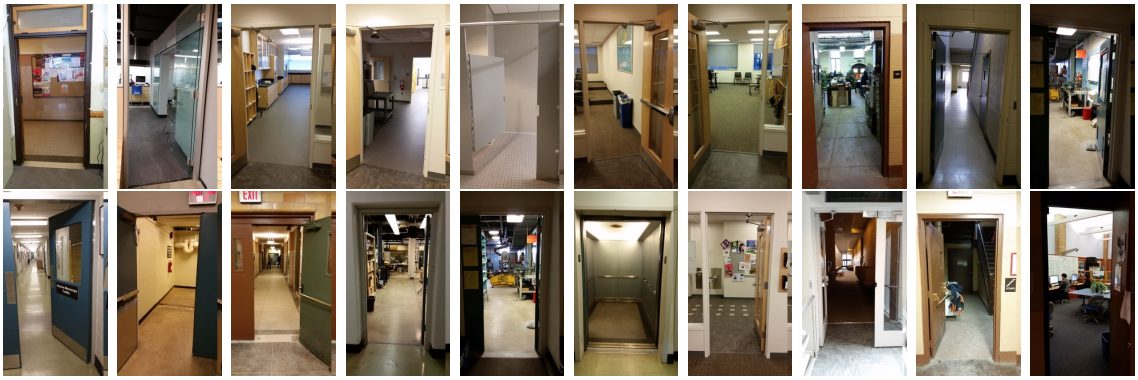

Figure 5.6: Examples of the variety of doors successfully traversed

In terms of "failure modes," the doorway detection system used in CoPilot is susceptible to false positives in that clustered vertical objects meeting the geometry constraints could be interpreted as doorways. For example, two tall file cabinets with a sufficient opening in between would be segmented as a doorway. However, while some may consider this a false positive, others might consider it a feature as it generalizes CoPilot to traversing a larger range of narrow openings. We should emphasize that since migrating to the occupancy histogram approach to doorway segmentation, no false positives have been observed when attempting an actual doorway traversal.

Finally, videos demonstrating the use of CoPilot can be found at `http://loveparkrobotics.com/?p=993` and `http://loveparkrobotics.com/?p=997`. The latter shows CoPilot integrated with a head array controller, an input device not well suited for the doorway navigation tasks. With the EPW in CoPilot mode and the doorway detected (i.e., when it puts the icon on the screen), a momentary tap of the rear switch embedded in the head-array will signal CoPilot to initiate door traversal. Just as with the Joystick mode of operation, the user can steal back control at any time by pushing the head-array switches.

5.7 Discussion

In this chapter we introduced CoPilot, an active driving aid that enables semi-autonomous, cooperative navigation of an EPW for automated doorway detection and traversal. The system demonstrated reliable navigation through a large population of representative doorways. The ability to simultaneously detect doorways and model the occupancy of obstacles in real-time were enabling factors to the robustness

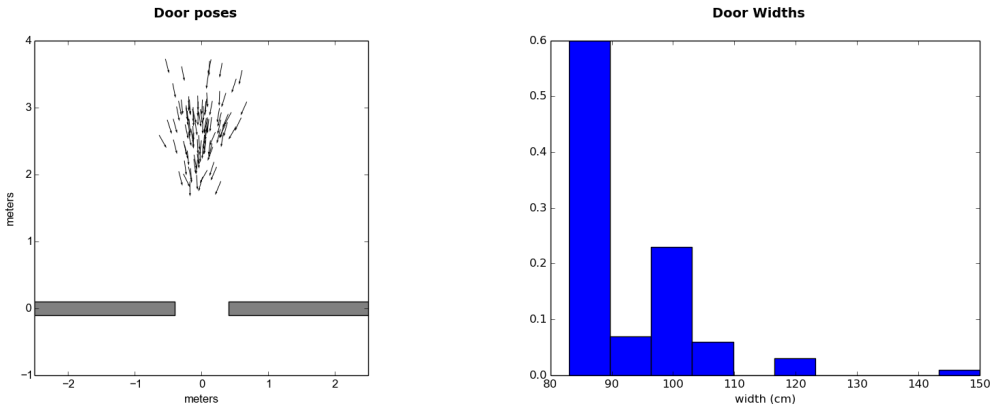

Figure 5.7: (Left) visualization of EPW starting poses with respect to a doorway centered at the origin with an orientation of -90° and (right) the probability mass function of the traversed door widths.

of the system. Long term autonomy was demonstrated in the sense that the system could repeatedly detect and traverse doorways. However, each doorway traversal was an independent navigation mission and the scale of the autonomous navigation task was limited to immediate perceptual neighborhood around the target doorway. In the next chapter, we extend this line of research into large scale outdoor urban environments.

Chapter 6

Service Robot Navigation in Outdoor Urban Environments

In the previous chapter (5), 3D point clouds were processed in real-time to both detect domain specific objects (doorways) and populate a local occupancy grid map. This processing was used for navigation goal selection and local path planning respectively. In this chapter, we broaden that line of research and present a system level approach to service robot navigation navigation in large scale outdoor urban environments. In addition to domain specific object detection and modeling the occupancy, here we use 3D point clouds to generate large scale landmark maps and real-time object detection to localize within those maps. The following sections describe the motivation of this approach, related works, an overview of the system, the map generation procedure, the navigation method, experimental results for both map generation and long term navigation, and a discussion.

6.1 Background

Smart wheelchair systems (SWS) have been an active research area for over 30 years [40]. The spectrum of work has ranged from component level safety sensors, to assistive controllers for steering, to completely autonomous solutions. This research interest is driven in part by their potential for improving the independence and quality-of-life of persons with disabilities and the elderly. Enabling mobility can allow individuals to participate more fully in basic activities such as employment, education, recreation, worship, commerce and other activities of community life that

most people take for granted. An additional positive side-effect of increased independence is significant cost savings in healthcare and long-term care [41]. While the large majority of research to date has focused on indoor operations, there is growing interest in extending SWS capabilities to outdoor environments [42, 43].

Our own approach to outdoor navigation for SWS was inspired by our work with autonomous automobiles [9]. By using a route network associated with a geographic coordinate system (e.g., latitude and longitude, UTM, etc), users merely specify a desired goal location and the automobile navigates there autonomously. This is made possible in large-part through the availability of accurate pose estimates (typically from high-performance GPS/INS systems), and exteroceptive sensors which provide rich three-dimensional (3D) data (e.g., Velodyne HDL-64E LIDAR) for robust perception in unstructured environments. Leveraging these sensor technologies, vehicles such as Google's driverless car have driven 100,000s of miles on public roads demonstrating performance at least as good as their human counterparts [44].

First, our emphasis is on navigation in urban environments. This is motivated by the observation that over 80% of the U.S. population resides in urban areas [45]. While the availability of GPS measurements in urban areas can typically be assumed, multi-path errors from buildings, trees, etc., can significantly compromise its accuracy. However, we further observe that these same structures can be used as landmark features to yield highly accurate relative position estimates. This is illustrated in a simple experiment shown at Fig 6.1. In this example, the SWS was manually driven around an 80 meter triangular-shaped sidewalk loop while logging GPS data. Portions of the path were lined with large trees. As a result, a large section of the path had significant position estimate errors due to multi-path. This is shown in the left satellite image. During the experiment, an Extended Kalman Filter (EKF) based SLAM algorithm was also executed. It took its initial pose from the GPS, but all subsequent pose estimates were derived from pole-like features (in this case, the trees) that were segmented by a LIDAR system. The resulting pose estimates are dramatically improved, as shown in the right satellite image. This motivates the use of feature-based SLAM algorithms to compensate for GPS errors. Furthermore, as with indoor paradigms, *a priori* maps embedding accurate landmark locations have the potential to significantly improve localization performance and robustness in urban environments.

Figure 6.1: A demonstration of the ineffectiveness of GPS as method of sidewalk level localization. (Left) The GPS signal is overlaid on satellite imagery while driving around the triangular-shaped path. (Right) The path derived from SLAM using trees as landmarks.

To this end, we employ a holistic approach to SWS navigation in urban environments. The proposed *SWS Ecosystem* has two primary components. The first is a mapping service which generates accurate, large-scale landmark maps and an associated route network that are made available through a cloud interface. The second is the SWS itself, which is a client of the mapping service. The SWS prototype integrates 3D LIDAR/imaging systems which provide robust perception in unstructured, outdoor environments. It also leverages these same sensors to perform map-based localization with a demonstrated accuracy at the decimeter level. The net result is an SWS platform with perception and localization systems suitable for autonomous navigation in urban environments. Furthermore, these capabilities are achieved at a monetary cost not prohibitive for the EPW consumer space.

The results presented herein extend our previous work in three significant ways. First, we developed and integrated a 3D LIDAR into the SWS prototype. This improved the safety and effectiveness of the system by enabling robust detection of both 3D landmarks and obstacles. Second, we developed a new vehicle for mapping at the sidewalk level. Using this, we created higher resolution and larger scale maps (3 times what was completed in the past), and also addressed the significant issue of sidewalk occlusion that we observed previously. Finally - and in part due to these

first two improvements - we have demonstrated autonomy over distances exceeding 12 km. This represents an order of magnitude improvement over our previous work.

6.2 Related Work

The proposed SWS Ecosystem has strong ties to research in smart wheelchair systems, large-scale urban mapping, and autonomous navigation of service robots in urban environments. While a complete survey of these topics is outside the scope of this document, we have tried to highlight at least the most relevant and recent works in each of these areas.

Smart wheelchair systems (SWS) have been an active research area since the early 1980s. Our own work in the SWS space spans over a decade, and includes [46–49]. A survey of the field (as of August 2005) can be found in [40]. More recent projects of note include the MIT Intelligent Wheelchair Project [50], the goal of which is to develop a voice-commanded autonomous wheelchair intended for use in indoor environments. The Home, Lift, Position, and Rehabilitation (HLPR) Chair [51] developed by NIST is a special-purpose assistive mobility device to provide independent patient mobility for indoor tasks, such as moving to and placing a person on a chair or bed. HLPR has demonstrated obstacle detection and navigation indoors with promising results. The Personal Mobility and Manipulation Appliance (PerMMA) [52] is being developed at the University of Pittsburgh and Carnegie Mellon University with the objective of combining manipulation and mobility assistance in support of complete independence for its users. The system employs two robotic arms, and has demonstrated object manipulation tasks such as retrieving a drink from a refrigerator.

In contrast to these other efforts, our work has been centered around outdoor systems. Our most current research, and the focus of this paper, emphasizes robust navigation in unstructured outdoor environments. Developing robust robotics solutions suitable for use outdoors is a significant challenge compared to indoor environments. The scale is much larger, illumination levels vary from strong sunlight to near complete darkness, the environment is far less structured, environmental conditions can change quickly and dramatically, and simplifying assumptions such as a level ground plane are not reliable. Furthermore, operations at the sidewalk level require localization performance beyond the bounds of what traditional GPS can provide. Since 2012, several other research groups have turned their sights towards outdoor

wheelchair systems as well. Yokozuka *et al.* [43] employ an approach similar to ours in that they actuate a 2D LIDAR to produce 3D point clouds of the environment. However, they use a 3D voxel grid as the map representation whereas we use a feature-based map to operate in similar outdoor urban environments. They report that their system has autonomously operated for at least two missions of over a kilometer during the Tsukuba Challenge whereas our system has performed autonomously for over 10km. Also, their approach to obstacle avoidance uses 2D information (the localization component uses the 3D data) which is not as robust as the 3D information used in our approach.

The work by Irie and Tomono [42] also has a wheelchair navigating an urban environment. Their approach uses maps that are already available and annotated for human use, say from Google maps, and use those to localize in urban environments. Their approach assumes that a grid map is available where grid cells have been labeled as belonging to a roadway, sidewalk, or building and localization is done by detecting the boundaries between these types of regions using stereo vision. This is similar to our approach in that 3D perception is used and that common urban features are used for localization. Key differences are in the chosen sensing modality and map representation. We deliberately ruled out a vision-based approach due to robustness concerns. They demonstrated successful position tracking on a 150m sidewalk course whereas our system demonstrated successful autonomous navigation (which assumes successful position tracking) of an order of magnitude more distance. Also, their assumption is that annotated maps of the environment will someday exist for easy consumption, but in this work every map had to be hand labeled. In this respect, they still require a mapping procedure.

One inspiration for our mapping approach is the Google mapping trike [53], which is used in areas where their mapping cars cannot traverse. Their platform has a high-end sensor suite with multiple sensing modalities whereas ours is a relatively low-cost platform. Our mapping approach reduces a 3D representation of the environment to a 2D planar map of features for the SWS client. Chong *et al.* [54] use a method whereby a 3D point cloud of the environment is created using a push broom mounting of a 2D LIDAR. They then project points of interest from the 3D point cloud to a 2D plane to create a synthetic 2D LIDAR scan that is invariant to roll and pitch of the mobile platform. They report successful localization on a 1.5 km route with quantitative performance similar to ours. Their localization approach uses a 2D occupancy

grid as a map representation whereas we use a feature based representation. They also demonstrate two successful autonomous missions on two shorter sub-routes. We have demonstrated both short term and long term autonomy in fifteen autonomous missions.

Mapping at the sidewalk level requires a sufficiently accurate 6 degree of freedom pose estimate in a global coordinate frame in GPS compromised areas. Baldwin *et al.* [55] use a push-broom style 2D LIDAR with the goal of providing an accurate pose estimate by using probabilistic methods on small swaths of aggregated push-broom scans. They also report better accuracy with their method in a GPS compromised outdoor environment than a reference high-performance GPS/INS system. Their localization approach uses previously seen swathes of the environment represented as point clouds. During operation of the localization system, the currently observed swathe is matched to the most likely swathe observed in a prior experience and the pose is estimated based on that match. They report successful localization results on 26km of roadway, but have not demonstrated autonomous navigation. We have not autonomously operated that amount of distance, but emphasize that we are operating at human scale, rather than road vehicle scale.

Lategahn *et al.* [56] map urban environments using predominately a stereo camera and IMU. Although, they use a GPS measurement to initialize the pose estimate. Similar to our work, they use 3D features as landmarks. However, their localization approach uses only a monocular camera and IMU in conjunction with the landmark map to provide an accurate 6 degree of freedom pose estimation. Our client system operates under a 3 degree of freedom pose assumption. Their system demonstrated a mean localization performance accuracy of 10cm on a 10km mapped urban environment. Our mapped environment is smaller, but our localization results are similar. Furthermore, while their work focuses on localization, our SWS Ecosystem offers a more complete solution for autonomous navigation.

Finally, an important aspect to navigation at the sidewalk level is the ability to handle dynamic obstacles in the environment such as pedestrians. Kummerle *et al.* [57] present a tour guide robot that autonomously navigated through a crowded 3.3 km urban route. They use a grid map representation of the environment and a particle filter based approach to localization. They demonstrate qualitatively successful localization results whereas we provide a quantitative analysis of localization

accuracy. Their approach relies on 2D information for localization and dynamic obstacle avoidance. While we are also concerned with dynamic obstacles, the emphasis of our work is in localization and mapping. Robustly handling dynamic obstacles is outside the scope of this work. However, our system utilizes 3D information for obstacle avoidance and can potentially perform better when there are low lying dynamic obstacles, such as dogs, which their system had trouble with.

6.3 SWS Ecosystem Overview

As alluded to earlier, the key for our SWS to navigate reliably in an urban environment is having an accurate landmark map and route network. The landmark map consists of the absolute locations of landmark features (in this case, pole-like features) and is used for localization purposes. The route network is used to indicate wheelchair accessible paths with respect to the landmarks and is used for path planning.

We approach the problem from a client service standpoint. Hence, our ecosystem is composed of two major systems: the SWS acting as a consumer of landmark maps and route networks, and a service platform used to generate landmark maps and route networks, and to make them available to SWS clients. The envisioned approach is to use a sidewalk-level mapping vehicle with high quality sensing equipment to enable a high fidelity 3D reconstruction of the environment. Then, the 3D reconstruction could be reduced to salient landmark features and their absolute locations with respect to a global coordinate frame. Route networks could then be generated via the service mapping platform or by manually driving the client SWS on safe paths in a learning phase before autonomous operation as in [46]. These landmark maps would then reside in the cloud, and the SWS would download the landmark map and route network based on its location and the desired goal locations chosen by the operator.

In this chapter, we describe a proof of concept implementation of the envisioned system. The mapping service platform and the map generation process is outlined in Section 6.4. The client SWS platform and methods it uses to perceive and navigate the environment are described in Section 6.5. Following the specifics of the service and client components, results of using the system in practice are discussed in Section 6.7.

Figure 6.2: Some example pole features. From left to right a street sign, parking meter, lamp post, and fire hydrant.

6.4 Server Side Map Generation

This section details the service component tasked with generating and providing the landmark map and route network. The map representation used by the system is feature-based, which was motivated by the need for the SWS to localize in a GPS compromised environment utilizing a relatively low cost sensor suite. The global feature map was generated by capturing and synthesizing a dense point cloud representation of the environment. Urban environments offer a plethora of features for tracking. In this work, we focused on pole-like features - herein referred to simply as *pole features* - such as parking meters, lamp posts, trees, etc. Examples of typical pole features in our map are shown in Fig. 6.2. In our previous work [49], we employed a street level mapping platform (a car) that had the advantage of utilizing a high-end OXTS RT-3050 GPS/INS system which provided accurate positional information. However, a shortcoming with mapping from the street was that features at the sidewalk level could be occluded by obstacles such as parked cars. This led to the construction of a sidewalk level mapping platform.

6.4.1 The Mapping Trike Platform

Taking inspiration from Google's mapping efforts [53], the Mapping Trike platform was built upon a commercial tricycle augmented with a sensor suite, depicted in Fig. 6.4. A Microstrain 3DM-GX3-45 inertial measurement unit (IMU) with GPS antenna and two 4096 cycles per revolution (CPR) resolution encoders were used to

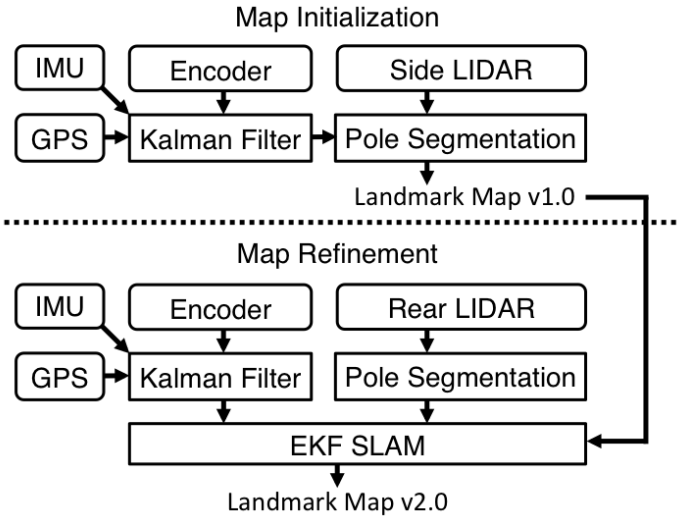

Figure 6.3: A block diagram representing the data flow of the mapping process. The section above the dashed line represents the map initialization phase and the section below the dashed line represents the map refinement phase.

provide pose estimation. Point data from two side-facing LIDARS (SICK LMS291-S14) and one rear-facing LIDAR (SICK LMS291-S05) were used to detect landmark features. The Mapping Trike was driven manually around an area of interest while logging data from all of its sensors. Data were then post-processed to synthesize the landmark map.

Synthesizing the map takes place in two main stages. The initialization stage creates a high fidelity 3D reconstruction of the environment via the side-facing LIDARS. This reconstruction is then reduced to landmark features that have an associated position and covariance. These landmark statistics serve as an initial estimate of the landmark map, which is later refined to correct positional errors of the landmarks.

6.4.2 Mapping Stage 1: Initialization

The initial map is generated in three main stages: pose estimation, landmark segmentation, and map synthesis. A data-flow diagram for the map initialization phase is depicted above the dashed line in Fig. 6.3.

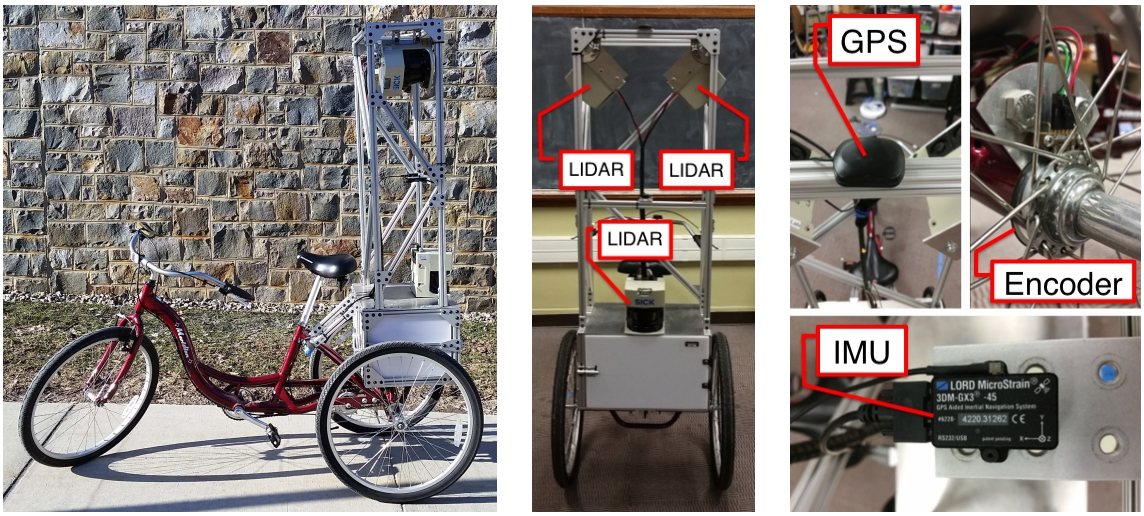

Figure 6.4: (Left) The Mapping Trike platform. (Center) A rear view of the Mapping Trike with mounting positions of the LIDARs indicated. (Right) Close up pictures of the GPS antenna, wheel encoder, and the Microstrain 3DM-GX3 IMU.

Pose Estimation

The localization module of the SWS works under the assumption that the landmark map is a 2D plane. Hence, the mapping goal is to register the landmarks to a consistent 2D global coordinate frame that is sufficiently planar in local neighborhoods. In our case, we use the Universal Transverse Mercator (UTM) coordinate system, which maps positions on the globe to a 2D Cartesian coordinate frame. To achieve this goal, the estimated pose of the trike with respect to the UTM frame must be accurate to within the operating tolerances of the SWS. The 3DM-GX3-45 has an integrated an on-board GPS, but its accuracy was insufficient to accurately register the trike to the UTM frame even when fused with inertial measurements from the IMU. Our solution was to integrate feedback from the wheel encoders and incorporate the kinematics of the trike into an extended Kalman filter (EKF). Thus, the trike pose was estimated by using the vehicle kinematics as the predictive step, and fusing the GPS and inertial measurements for the corrective step. We used a predictive motion model of the

form:

$$\mathbf{x}_{k+1}^{-} = \begin{pmatrix} x \\ y \\ z \\ \alpha \\ \beta \end{pmatrix}_{k+1} = \begin{pmatrix} x \\ y \\ z \\ \alpha \\ \beta \end{pmatrix}_{k} + \begin{pmatrix} \cos\alpha\cos\beta & 0 \\ \sin\alpha\cos\beta & 0 \\ \sin\beta & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta_l \\ \Delta_a \end{pmatrix}, \tag{6.1}$$

where $\mathbf{x}$ denotes the state vector consisting of the Cartesian coordinates (x, y, z) and the yaw and pitch angles (α, β), Δ_l and Δ_a were the linear and angular displacements over some small time step derived from the encoders. The k subscript denotes a time step and the superscript $^-$ denotes the prediction of the state and covariance after the motion update. The roll angle was not tracked because the effect of roll was minimal on flat sidewalks, but tracking the pitch was crucial as small changes in pitch could distort the projection of the rear-facing LIDAR data. For example, the map in Section 6.7 has hills with a grade of approximately 8.5%.

For the purposes of the EKF, the covariance P was updated as:

$$P_{k+1}^{-} = J_s P_k J_s^T + J_m Q J_m^T, \tag{6.2}$$

where J_s was the Jacobian with respect to the state defined as:

$$J_s = \begin{pmatrix} 1 & 0 & 0 & -\Delta_l \sin\alpha\cos\beta & -\Delta_l \cos\alpha\sin\beta \\ 0 & 1 & 0 & \Delta_l \cos\alpha\cos\beta & -\Delta_l \sin\alpha\sin\beta \\ 0 & 0 & 1 & 0 & \Delta_l \cos\beta \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.3}$$

J_m was the Jacobian with respect to the linear and angular displacement defined as:

$$J_m = \begin{pmatrix} \cos\alpha\cos\beta & 0 \\ \sin\alpha\cos\beta & 0 \\ \sin\beta & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \tag{6.4}$$

and Q is additive Gaussian noise in the linear and angular displacements.

The 3DM-GX3-45 streams the GPS data and IMU data at different rates, 4 Hz and 100 Hz respectively. So, in our EKF formulation each of these data streams has

a separate measurement update. The GPS measurement update is computed with the following sequence of equations:

$$\begin{aligned} K_{g,k} &= P_{k+1}^{-} H_g^T (H_g P_{k+1}^{-} H_g^T + R_g(k))^{-1} \\ \mathbf{x}_{k+1} &= \mathbf{x}_{k+1}^{-} + K_{g,k}(\mathbf{z}_g - H_g \mathbf{x}_{k+1}^{-}) \\ P_{k+1} &= (\mathbf{I} - K_g H_g) P_{k+1}^{-}, \end{aligned} \tag{6.5}$$

where $\mathbf{z}_g$ is the GPS measurement represented as a UTM coordinate, $\mathbf{x}$ is the state vector, and H_g is a projection matrix to extract the x and y Cartesian coordinates from the state vector defined as:

$$H_g = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}. \tag{6.6}$$

$R_g(k)$ is a covariance matrix that represents the uncertainty of the GPS measurement at time step k, $\mathbf{I}$ is the identity matrix, $\mathbf{x}_{k+1}$ and P_{k+1} are the corrected state and covariance respectively. The IMU itself does some on-board filtering and the covariance of the GPS measurement can be requested. This is the value used for $R_g(k)$.

Similarly, the gyro measurement update was computed with the sequence of equations:

$$\begin{aligned} K_{a,k} &= P_{k+1}^{-} H_a^T (H_a P_{k+1}^{-} H_a^T + R_a(k))^{-1} \\ \mathbf{x}_{k+1} &= \mathbf{x}_{k+1}^{-} + K_{a,k}(\mathbf{z}_a - H_a \mathbf{x}_{k+1}^{-}) \\ P_{k+1} &= (\mathbf{I} - K_{a,k} H_a) P_{k+1}^{-}, \end{aligned} \tag{6.7}$$

where $\mathbf{z}_a$ is a gyro measurement of the form $[\alpha, \beta]^T$, and H_a is a projection matrix to extract the yaw and pitch from the state vector defined as:

$$H_a = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6.8}$$

$R_a(k)$ is a covariance matrix that represents the uncertainty of the gyro measurement at time k, $\mathbf{I}$ is the identity matrix and $\mathbf{x}_{k+1}$ and P_{k+1} are the corrected state and covariance respectively.

Landmark Segmentation

The EKF described above provided an acceptable pose estimate for the mapping trike. The next step was to register the LIDAR scans to a common coordinate frame (in our case, the Standard UTM frame), as we are relying upon the motion of the trike to

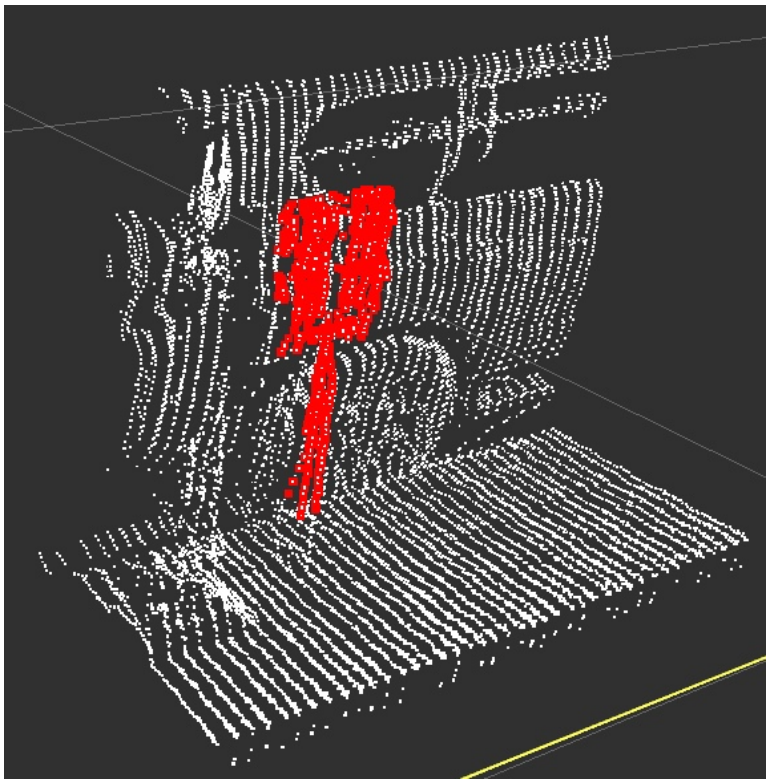

Figure 6.5: Pole segmentation of a parking meter from the dense 3D reconstruction of a side facing SICK LIDAR.

build the 3D reconstruction of the sidewalk scene via 2D laser scans. Because of this, in our data collection we aimed for a speed of approximately 1 m/s. In conjunction with the LMS291 LIDAR scan rate of 75 Hz, this gave us a vertical scan for each 1-2 centimeters of distance traveled.

Once we had a sufficiently dense set of LIDAR scans registered to a common frame, we segmented the pole features by aggregating subsequent 2D scans into a 3D point cloud. The scans were aggregated in a sliding window fashion. The first n scans made up the first point cloud and subsequent point clouds began at the middle scan of the previous point cloud. The reason for this overlap was that a landmark could be missed if it straddles the boundary of the two windows. Additional bookkeeping was done to eliminate the duplicate landmarks in the overlapping regions. For each window, the points corresponding to the ground plane were removed using a RANSAC procedure. The remaining points were then clustered based on the Euclidean distance to neighboring points. The maximal intra-cluster distance was set to 10 cm. An example of the segmentation of a parking meter from a window containing 100 scans

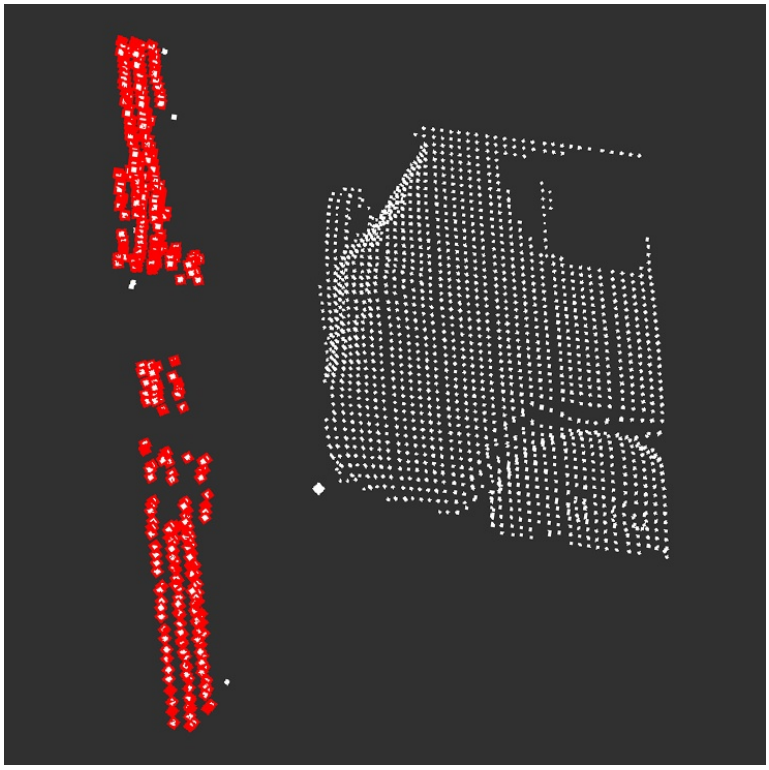

Figure 6.6: An example of a lamp post that would be clustered into two components due to a section of invalid measurements.

is shown in Fig. 6.5.

Unfortunately, some features were clustered into several disjoint clusters using this approach. This was most likely to occur when observing lamp posts due to the low albedo surface. An example of a lamp post that would be improperly segmented is shown in Fig. 6.6. Based on the observation that vertical features are the focus of our landmark segmentation, we performed a cluster merging step to combine clusters that were likely to belong to the same vertical object. The centroids of each cluster were projected to the $x - y$ plane. If the Euclidean distance between two centroids was less than 20 cm, the clusters were merged into a single cluster. This merging phase significantly increased the accuracy of the entire segmentation process.

The next step was to validate that a cluster was a pole feature by passing some validation gates. The approach of fitting a cylindrical model, while intuitive was not very effective. This was especially true for features like street signs and parking meters which lacked uniformity. That being the case, we validated a cluster's status as a pole by first performing a spectral decomposition of the covariance matrix of its data points. This gave us information that was invariant to the coordinate system in

which the data was measured. The eigenvector associated with the largest eigenvalue λ_1 was compared against the vector $[0, 0, 1]^T$ (corresponding to the gravity vector). If the angle between them was outside some tolerance, the cluster was rejected as a landmark; a cluster passing this validation gate was assumed to be a tall vertical object. We further examined the aspect ratio of the cluster. Since the eigenvalues correspond to the variances in the principal directions and the largest direction was already determined to be vertical, several ratios of the eigenvalues were tested: $\lambda_2/\lambda_3 \approx 1$ to validate that the width and depth were similar and $\lambda_1/\min(\lambda_2, \lambda_3) > 2$ to validate the object was at least twice as tall as it was wide. Only after passing these gates was a cluster considered a candidate landmark.

Landmark Map v1.0

Ultimately, the landmarks associated with each cluster needed to be transformed to 2D UTM coordinates. The reason for this is that the client SWS is not assumed to have the sensing capability to track its position in $\mathbb{R}^3$. As such, the trike pose and centroid of each cluster were transformed to UTM coordinates and each landmark cluster was reduced to a feature vector of the form $[\rho, \phi, s]^T$ where ρ and ϕ were the range and bearing to the transformed landmark's centroid with respect to the trike's transformed pose at which the landmark was detected and s was a *signature* for the landmark (in our case the radius of the pole). The signature value aids in data association in the map refinement step (§ 6.4.3) and was computed by finding the bounding box of the landmark cluster's data points and returning $min(width, depth)/2$ as the radius.

For our landmark map, we wish to embed both the mean position and covariance of each landmark in the UTM frame. Using the 2D projection of the trike's pose estimate, denoted as $\mathbf{x} = [x, y, \theta]^T$, we converted the landmarks to the UTM frame as follows:

$$\mu_L = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} x + \rho\cos(\phi + \theta) \\ y + \rho\sin(\phi + \theta) \end{pmatrix}, \tag{6.9}$$

where μ_L denotes the mean location of a landmark. To compute the covariance we need to transform the noise model of the LIDAR to Cartesian coordinates. Note that the variance of the signature is not considered here as it does not depend on the

position. The noise model is denoted as:

$$Q = \begin{pmatrix} \sigma_\rho & 0 \\ 0 & \sigma_\phi \end{pmatrix}, \tag{6.10}$$

and is linearized with the Jacobian of μ_L with respect to ρ and ϕ, defined as:

$$H_Q = \begin{pmatrix} \cos(\phi+\theta) & -\rho\sin(\phi+\theta) \\ \sin(\phi+\theta) & \rho\cos(\phi+\theta) \end{pmatrix}. \tag{6.11}$$

However, we also need to propagate the uncertainty in the trike's pose into the uncertainty in the landmark's position. The linearization is performed using the Jacobian of μ_L with respect to x, y, and z, defined as:

$$H_t = \begin{pmatrix} 1 & 0 & -\rho\sin(\phi+\theta) \\ 0 & 1 & \rho\cos(\phi+\theta) \end{pmatrix}. \tag{6.12}$$

The covariance for a landmark's position is then

$$\Sigma_L = H_t \Sigma_t H_t^T + H_Q \Sigma_Q H_Q^T. \tag{6.13}$$

The initial map estimate is the set of means and covariances for each landmark feature denoted

$$M_{v1} = ((\mu_{L1}, \Sigma_{L1}), \ldots, (\mu_{LN}, \Sigma_{LN})) \tag{6.14}$$

where N is the number of landmarks.

6.4.3 Mapping Stage 2: Refinement

In our previous work [46], the mapping vehicle employed a high performance OXTS RT-3050 GPS/INS system for pose estimation. Despite this luxury, the residual errors in initial landmark positions were large enough to make SWS localization impossible. As a result, a second mapping stage was required to refine the landmark positions. Furthermore, the 3DM-GX3-45 used on the Mapping Trike lacks the accuracy of the OXTS RT-3050, and so a map refinement step was expected.

The map refinement stage used an EKF simultaneous localization and mapping (SLAM) approach modified to use M_{v1} as an input. While the map initialization stage was performed off-line using the trike's side-facing LIDARs, the map refinement stage was performed on-line using trike's rear-facing LIDAR. Extracting poles from the rear-facing LIDAR was done by first registering the laser scan to the trike's coordinate

Figure 6.7: A comparison of an initial landmark map to the refined map. The trike started on the left with an acceptable GPS position estimate and traveled to the right up a grade of approximately 8.5%. The red triangles indicate the landmarks in M_{v1} and the yellow circles indicate the landmarks in M_{v2}. A landmark at each corner was tagged with satellite imagery positional information to act as known correspondences.

frame. The point data in the scan were then clustered based on the Euclidean distance of neighboring points; the maximal intra-cluster distance was set to 10 cm. This value was chosen to capture pole features at a range of approximately 8 meters. Any cluster of less than 4 points was rejected as a possible landmark. A circle model was then fit to each remaining cluster using RANSAC [58] and the model parameters were refined using a least squares fit to the inliers. Any cluster that had less than 90% inliers (a measure of model fit) or a fitted circle radius greater than 40 cm (none of the landmarks have a radius this large) was rejected as a landmark. Each accepted landmark was then put into the $[\rho, \phi, s]^T$ form by converting the center point of the fitted circle to polar coordinates with respect to the trike. The radius of the circle was used for the signature s. Note that the center of the fitted circle was used rather than the centroid of the cluster's data as it was a better approximation of the landmark's actual location when seen from different viewing angles.

As in a typical EKF SLAM implementation, the vehicle pose and map locations were represented as a Gaussian with mean vector $\mu = [\mathbf{x}, \mathbf{m}_1, \ldots, \mathbf{m}_n]^T$ where $\mathbf{x} = (x, y, \theta)$ was the trike's pose and $\mathbf{m}_i = (x_i, y_i, s_i)$ was the i^{th} landmark's position and

signature, and covariance, defined as:

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm_1} & \cdots & \Sigma_{xm_n} \\ \Sigma_{m_1x} & \Sigma_{m_1m_1} & \cdots & \Sigma_{m_1m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_nx} & \Sigma_{m_nm_1} & \cdots & \Sigma_{m_nm_n} \end{pmatrix}, \tag{6.15}$$

where Σ is a block matrix and each block corresponds to the covariance of the pose Σ_{xx}, the pose with respect to landmark Σ_{xm_i}, or the covariance of a landmark $\Sigma_{m_im_i}$.

The first step to our map refinement is to pre-process M_{v1} by tagging a small number of specific landmarks with accurate position information using satellite imagery. This information provides known correspondences for these features and aids in loop closure. We found this necessary, as the loops we are considering are on the order of a kilometer in length. The SLAM measurement update step computes the Mahalanobis distance from an observation to each landmark in M_{v1}, as well as to each landmark in the refined map maintained by the EKF denoted as M_{v2}. The maximum likelihood correspondence (MLC) [26] is selected and the update has three possible outcomes:

1. The MLC is below some threshold and the observation is discarded.

2. The MLC is associated with a landmark in M_{v2} and the EKF is updated normally.

3. The MLC is associated with a landmark in M_{v1} and the landmark feature is removed from M_{v1} and added to M_{v2}. Additionally, if the landmark has associated satellite data, then that position is used when adding the landmark to M_{v2}.

The refined landmark map v2.0 is the result of the EKF SLAM procedure. Fig. 6.7 shows the difference between one instance of the initial map compared to the refined map on Webster Street. The trike started on the left with an acceptable GPS position estimate and traveled to the right up a grade of approximately 8.5%. The red triangles indicate the landmarks in M_{v1} and the yellow circles indicate the landmarks in M_{v2}. A single landmark at each corner was tagged with satellite imagery positional information to act as known correspondences.

6.4.4 Route Network Generation

The SWS uses a route network 4.1.2 for global path planning containing information about wheelchair accessible paths with respect to the landmark map. Each edge also has semantic information associated with it: a speed limit, a stop condition, and a weight. The speed limit is used to constrain the local path planner (§ 4.3.3) to a maximum speed in areas of the map where high speed traversal could be unsafe. The stop condition is used in areas of the map, such as street crossings, where the operator of the SWS must determine when it is safe to cross. The edge weight is used by the global planner to search for paths that minimize the expected time of arrival.

The route network was constructed by first sampling the trike poses as corrected by the map refinement step. The trike was driven on acceptable wheelchair accessible paths in order to ensure that the sampled path is valid for the SWS. After the initial route network was constructed, semantic information associated with the edges was added manually. While not fully automated, the associated workload is not too cumbersome. For example, the route network used in our experiments (§ 6.7) only had to have the six street crossings annotated by hand.

6.5 The Smart Wheelchair System (SWS) Client

This section describes the operation of the SWS under the assumption that the landmark map and route network are available. The SWS software has three major components: perception, localization, and navigation. The perception component of the SWS has two primary tasks: pole feature segmentation for localization, and populating a cost map for obstacle avoidance. The localization component maintains an estimate of the SWS with respect to the landmark map by utilizing the output of the pole feature detection process. Finally, the navigation component uses the route network for global planning and the cost map (populated via the 3D sensors) to generate local trajectories. A detailed description of each component, as well as the SWS hardware platform, now follows.

6.6 The SWS Platform

The SWS used in this work is pictured in Fig 6.8. It is an Invacare M91 Pronto that integrates the FDA approved Motion Control Module (MCM) developed under

Figure 6.8: The SWS prototype with the fields of view of the actuated LIDAR (blue) and each 3D camera (orange) highlighted. (Left) A profile view and (Right) a top down view of the respective sensors' fields of view.

our previous work [47]. The MCM provides a seamless interface for regulating SWS linear and angular velocities. Proprioceptive sensing includes high resolution quadrature encoders (4,096 CPR) and a Microstrain 3DM-GX1 IMU. The latter is used to enhance the odometry performance by providing gyro corrections. Exteroceptive sensors include two IFM O3D200 3D cameras 2.4.1 and an actuated Hokuyo UTM-30LX 2D LIDAR (described in detail in Section 2.4.2). These provide the necessary 3D sensing for robust obstacle detection and landmark segmentation. The total cost of all sensors is <$10K USD in quantities of one.

The primary role of the IFM O3D200 3D cameras was to detect obstacles in the environment and propagate that information to the local map (§ 4.3.1). In our previous work [49] the 3D camera was crucial for robust obstacle detection as it was the only 3D sensing modality used. The current SWS integrates a second 3D sensing modality (§ 2.4.2), but the O3D200 is still useful for ground plane segmentation and to cover the near field and low-lying blind spots. In addition, to compensate for the limited field of view, two O3D200 cameras are used. Fig. 6.8 shows the mounting positions and depicts the vertical and horizontal fields of view for each sensor.

The SWS also features an actuated Hokuyo UTM-30LX LIDAR, herein referred to as the 3D Hokuyo. Although it was also used for obstacle detection, the primary role of the 3D Hokuyo was landmark segmentation. Our motivation for actuation was due to robustness concerns in our previous work [49]. Specifically, localization with the UTM-30LX was done using 2D information, but the SWS operates in a 3D environment. While this was sufficient for demonstration purposes, robust segmentation

of 3D landmarks requires a 3D sensor.

The software was developed using the Robot Operating System (ROS) [27] framework and modularized based on the message passing paradigm used by ROS. Additionally, the Point Cloud Library (PCL) [59] was leveraged for processing point cloud data from the exterioceptive sensors. The computational processing for the software components was performed by a laptop with an Intel Core i7-2760QM 2.4GHZ processor and 4 GiB of RAM.

6.6.1 Ground Plane Tracking

Reliable obstacle detection and mapping requires the local ground plane to be tracked. For the SWS, this is accomplished using the O3D200 3D cameras and employing an iterative reweighted least squares (IRLS) approach. IRLS has advantages over a traditional RANSAC approach in that it integrates both temporal filtering as well as regularization through the use of a calibrated ground plane. In [9], we found that it outperformed RANSAC for road segmentation, and therefore employed it here as well.

IRLS starts with an estimate of an ideal ground plane derived from the extrinsic calibration parameters of the 3D cameras. We model the plane at time k as

$$\Pi_k = a_k x + b_k y + c_k z + dk = 0. \tag{6.16}$$

The assumption is that the ground plane orientation changes with time, but that the rate of change is small in comparison to the scan rate of the 3D camera ($\approx$ 7 Hz). Let $P_k \in \mathbb{R}^{3\times n}$ denote the n points returned from a 3D camera image at time k. The normal distance from point $\mathbf{p}_k = [x_k, y_k, z_k]^T \in P_k$ on the ground plane at time k to the estimate to the ground plane at the previous time Π_{k-1} should be small in practice. This notion is formalized by solving a problem of the form:

$$\begin{aligned}\min_{a,b,c,d} \sum_{i=1}^{n} W(\mathbf{p}_{k_i}, \Pi_{k-1})(ax_{k_i} + by_{k_i} + cz_{k_i} + d)^2 + \\ \sum_{j=1}^{m} W(\mathbf{q}_j, \Pi_{k-1})(ax_j + by_j + cz_j + d)^2,\end{aligned} \tag{6.17}$$

where $\mathbf{p}_{k_i}$ denotes the i^{th} point of the 3D camera image at time k, $W : \mathbb{R}^3 \times \Pi \to \mathbb{R}$ is a weighting function of the form:

$$W(\mathbf{p}, \Pi) = \left[\frac{1-\alpha}{1 + \beta\ \exp(\gamma\ f(\mathbf{p}, \Pi) - \delta)} + \alpha\right] g(\mathbf{p}), \tag{6.18}$$

where the first expression is a logistic function of the normal distance from the point $\mathbf{p}$ to the plane Π denoted as $f(\cdot)$ and the second expression is a function $g(\mathbf{p})$ that scales the measurement based on the x component of $\mathbf{p} = [x, y, z]^T$. This is to mitigate the effect of a greater density of points returned at closer ranges. The first term in the minimization problem operates on the points in the current 3D camera scan P_k. The second term is a regularization component that operates on points $Q \in \mathbb{R}^{3\times m}$ that are uniformly sampled from the ideal ground plane.

6.6.2 Landmark Segmentation

SWS landmark segmentation follows the same general procedure described in Section 6.4.2, but data from the 3D Hokuyo are used as input. These data have different characteristics than the side-facing LIDAR data of the Mapping Trike. First, the ground plane does not need to be segmented as it is first seen at approximately 9 meters, and we restrict landmark segmentation to ranges of 8 meters or less. Second, the vertical angular resolution is larger with the actuated LIDAR, so the clustering procedure uses a maximal intra-cluster distance of 45 cm rather than the 10 cm figure previously stated.

6.6.3 SWS Navigation

At the user level, navigation only requires two inputs. The first is selecting a destination. The second is resuming the SWS when it pauses at locations where the user must determine when it is safe, such as crossing a street. The latter is necessary, as some street crossings feature high speed automobile traffic that would be detected too slowly with the on-board sensor suite.

Localization of the SWS was performed using the particle filter approach described in Section 4.2. The particle filter approach was chosen because it was more robust to aspects of the environment that were not modeled by the system, namely the assumption the SWS is operating on a perfect 2D plane. The motion noise parameters were empirically determined under the assumption that a diverse particle set improves localization performance. As such, the values chosen ($a_1 = 0.05, a_2 = 0.01, a_3 = 0.001, a_4 = 0.1$) were an exaggeration of the true noise model. For this application, the signature value used to enhance data association was the radius of the pole feature. The number of particles used in our implementation was 60 and, for the purposes of

planning, the mean of all the pose components was used.

The global planner for the SWS is intuitive; given the waypoint network, $G(V, E)$, described in Section 6.4.4 and a desired destination $v \in V$, the path to the destination is computed via Dijkstra's algorithm. Note that the weight for each edge is the estimated traversal time rather than the distance, so the path returned from Dijkstra's algorithm minimizes the estimated traversal time and does not necessarily correspond to the shortest distance.

For local planning, the sample based approach on the input space of the linear and angular control velocities (v, ω) described in Section 4.3.3 was used. The range of velocities sampled for this application was $v \in [0.1, 1.2]$ m/s, and $\omega \in [-0.3, 0.3]$ rad/s. Updates to the costmap were performed asynchronously whenever a scan from any of the sensors was available, with an objective feedback rate of ≈ 7 Hz. The costmap was centered on the SWS and covered an area of 6×6 meters with a grid cell resolution of 5 cm.

6.7 Experimental Results

To demonstrate the effectiveness of the two major components of the smart wheelchair ecosystem, namely the server mapping component and the client SWS, a map was constructed following a path starting at the loading dock of Packard Lab and then going across the street and circling a multi-block loop in South Bethlehem, PA. This route (shown at Fig. 6.9) was chosen for several reasons:

1. The route was of significant length (approximately 1 km) and had diversity in the frequency and signature of landmarks.

2. The route contained a large loop which allowed us to test loop closure.

3. The route was not flat. The streets running in the north-south direction had a grade of approximately 8.5%. This allowed us to test the effectiveness of the 2D localization and mapping approach across significant changes in elevation.

We should emphasize that the density of landmarks played no role in the selection of this route, as pole features appear to be universally pervasive in South Bethlehem.

The following sections describe the mapping results and SWS navigation results.

Figure 6.9: Satellite view of the block route network. The route is depicted as the green path. The yellow circles indicate the landmark features. The distance around the loop is approximately 944 meters. The six stop sign icons correspond to locations where the SWS automatically stops and waits for the operator to determine when it is safe to cross. The numbered locations correspond to labeled destinations for SWS navigation.

6.7.1 Server Mapping Performance

To test the performance of the server mapping component, we mapped a multi-block area in South Bethlehem, PA shown in Fig. 6.9. The route is approximately one kilometer in length, and features a loop of approximately 944 meters. To validate the effectiveness of landmark segmentation by the Mapping Trike, the number of landmarks along this route (187) were counted by hand as a ground truth measure. This was compared against the output of the map initialization and refinement stages. The landmark segmentation procedure in the map initialization step successfully detected 186 of the ground truth poles. The sole missing landmark was a lamp post with a bike chained to it, so its omission was not unexpected. More significantly, there were 32 false positives which were associated with stationary pedestrians and the corners of buildings. The latter occurred when the sliding window only caught the very beginning of a building facade. These false positives could be readily eliminated by not

accepting landmarks at the leading edge of the sliding window, as they would reappear in the center of the window at the next time-step and be easily discriminated. Eliminating false positives from pedestrians would be somewhat more complicated, likely requiring a vision system implementing people detection as a validation gate. As this was not available in the current Mapping Trike configuration, these false positives were removed manually.

By comparison, the final landmark map contained 185 landmarks after the refinement stage. The sole discrepancy between this value and the ground truth was due to a false data associations in the EKF SLAM procedure where two poles were sufficiently close to one another (< 10cm) and associated as being the same pole. As a measure of map accuracy, the final landmark map was compared to the satellite image. The coordinates of 24 clearly visible poles were obtained by clicking their positions on the satellite map and the distances to the associated poles in the landmark map were computed. The mean error in distance was 66cm with a maximum error of 115cm and a minimum error of 28cm. These values are only an approximate measure of map accuracy as errors in clicking and satellite imagery are confounding factors.

To validate the generality of the server mapping component, we mapped a second neighboring block depicted in Fig. 6.10. The reason for this was that the parameters for the mapping procedure (e.g., the pole segmentation parameters) were tuned using data from this same loop. As a result, a test set disjoint from the training set was required. Using the same parameters as the first map, all 129 landmarks in the second map were successfully detected. However, 25 false positives were also detected. In addition to the types detected in the first map, there were also false positives from the supports of windows and glass doors, as well as from the fence posts of a chain link fence. These additional false positives are interesting cases. To elaborate, like the mapping vehicle our SWS segmented landmarks using 3D point cloud data. As a result, window supports or chain link fence posts likely would also be detected as landmarks by the SWS and correctly associated with those in the map. So, an argument could be made for labeling these as true positives and keeping them in the map. However, an alternate sensing modality (e.g., a camera based vision system) likely would not detect these as landmarks. Ultimately, since they did not meet our strict geometric definition for a pole feature, they were categorized as false positives.

We also note that the north-south distance of both loops was approximately 175

Figure 6.10: Satellite view of a second block mapped with the trike using the same parameters as the first block. The red circles indicate the landmark features and the yellow circles indicate false positives. The distance around the loop is approximately 585 meters. Out of 129 manually counted ground truth poles 100% were successfully segmented. However, 25 false positives were detected.

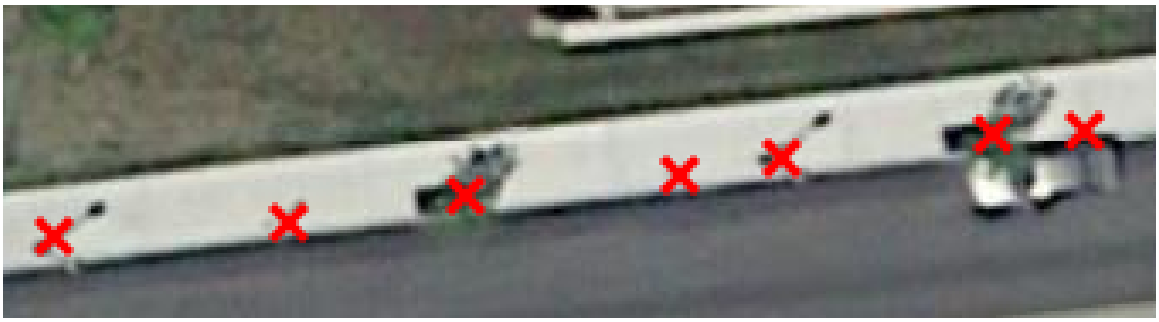

Figure 6.11: A close up view of the map in Fig. 6.9. The landmarks are depicted as red crosses.

meters along a grade of 8.5%. Without compensating for this grade, landmark positions on the north and south ends of the map would be shifted 60-70cm which would place many in the street. However, a review of the map indicates that these did in fact remain on the sidewalk. This validated our approach to projecting the landmark positions to the UTM frame. Since it is difficult to qualitatively discern the landmark accuracy at the scale of the map in Fig 6.9, Fig. 6.11 depicts a zoomed in view of a section of the map in 6.9. This section was chosen for two reasons: the landmarks are visible in the satellite image and it is along the east-west direction and could suffer from the previously mentioned shifting effect.

6.7.2 SWS Localization Accuracy

In an attempt to quantify the accuracy of the proposed map-based localization approach, the SWS was manually driven around the loop but with the localization module running. It was then stopped immediately adjacent to a reference landmark, and the normal distance from the base of the SWS to the landmark was measured manually. At the same time, this distance was also captured using the SWS's localized position. This process was repeated for a total of 30 landmarks.

This experiment was conducted a total of three times with landmark maps containing 50%, 75%, and 100% of the total landmarks. The motivation for reducing the number of landmarks available to the localization module was to assess the impact of landmark occlusion, e.g., what happens if half the landmarks are occluded by pedestrians? The results of these experiments are summarized in Fig. 6.12 with box plots for each landmark map. These indicate that sub-decimeter level 1D accuracy could be expected when even just half the landmarks are visible. To place these results in

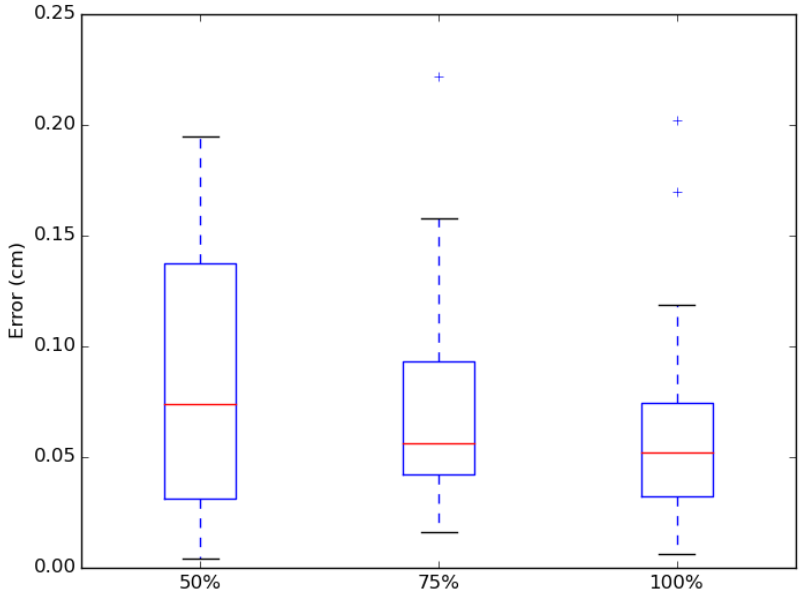

Figure 6.12: A comparison of 1d localization error from 30 reference locations. Localization was performed on three landmark maps containing 50%, 75%, and 100% of the landmarks. Each respective box plot shows the median, 25th, and 75th percentiles, with outliers plotted as individual points.

the proper context, we note that the authors in [55] evaluated their own approach in a somewhat urban environment against a "high caliber" DGPS/IMU system. They achieved a median path error of ≈ 0.5 m, while the DGPS/IMU error was ≈ 1.0 meter. While theirs and our test procedures were not entirely consistent, these results support the assertion that our map-based localization approach can achieve significantly better performance than a DGPS/IMU solution in urban environments.

6.7.3 SWS Navigation Performance

To demonstrate autonomous navigation of the client SWS, the landmark map and the route network generated from § 6.7.1 were downloaded by the SWS from the cloud to support the navigation task. Subsequently over the course of two days, the SWS drove autonomously around the loop (a distance of approximately 944 meters) five times in each direction. The reason for 10 loops was to demonstrate a reasonable level of reliability. The rationale for both clockwise and counterclockwise loops was not just scene diversity. We also wanted to investigate the impact (if any) of bias in

Figure 6.13: Changes to the environment not modeled by the system. (Left) winter conditions raised the curb cut out by 8cm. (Center) a parking meter covered by a bag. (Right) A bike chained to a tree.

the mapping procedure, as the map was constructed with the trike driving in only one direction. In our test protocol, once initiated the operation of the SWS was to be completely autonomous. The only exception was when groups of pedestrians blocked the sidewalk, where it was manually paused for safety considerations. With lone pedestrians, the SWS was not hindered from performing obstacle avoidance.

Testing was run under winter conditions that were less than ideal. Fig. 6.13 depicts some of these instances. In some areas, frost heave significantly disturbed sidewalk pavers. This caused surface deviations as much as 8 cm in height. Generous use of snow melt (i.e., rock salt) further reduced traction in these areas. Despite these challenges, the SWS motor controller performed well, although there was a noticeable decrease in velocity when traversing up steep grades. Landmarks in the map were also changed in unexpected ways. For example, a sequence of ten parking meters featured in the landmark map were covered by local law enforcement to indicate that parking was temporarily prohibited (Fig. 6.13). These coverings altered the signatures reported by the 3D Hokuyo which the localization relies upon.This particular stretch has two of the largest dead reckoning lengths ($\approx$ 13.5 meters) where the only landmarks were these parking meters. In spite of these discrepancies, accurate localization was maintained.

In total, the SWS traveled a distance of 10.3 km over an operational time of 222 minutes. The longest run of continuous operation was for 88 minutes covering 3.7 km, and was only suspended due to depletion of the laptop battery. No localization failures were observed during testing. However, during one trial manual intervention was required due to a control "failure." Specifically, a large amount of road salt in

Figure 6.14: Photos of the SWS in operation, highlighting interactions with pedestrians (left), the narrowness and clutter of certain sidewalk areas (center), and automatically pausing at a crosswalk (right).

an area where sidewalk pavers were pushed up significantly from frost heave (similar to Fig. 6.13) resulted in the SWS drive wheels spinning in place, and the SWS could not overcome the "obstacle" on the first try. This required the operator to manually back up the SWS less than 1 meter. A second attempt under autonomous control was successful and the trial continued to the end. We should emphasize that localization was not lost even with the wheel slippage as the 3D Hokuyo was able to observe landmarks during the episode.

Additional images highlighting points of interest during course navigation are shown at Figure 6.14. A more informative video from testing can be viewed at `http://vader.cse.lehigh.edu/videos/sws_navigation.mpeg`.

An interesting question posed by one of the reviewers (for which we are grateful) was the impact on system performance if false positives in the landmark map had not been removed. We had assumed this was necessary, but was this in fact the case? To assess this, logged data from the ten trials was used to localize the SWS using the landmark map with both true and false positives in place. If a localization failure occurred during playback of the log file, the SWS was manually re-localized and the log file was continued from that point. Of the five clockwise trials, three had localization failures. Two suffered from a single failure but completed the loop upon resumption. The third had to be reset twice. Of the five counter-clockwise trials, two had localization failures. One was able to complete the loop after being manually reset, while the second had to be reset twice. Thus, the success rate for this route dropped from 100% to 50%, motivating the need for an accurate landmark map.

6.7.4 SWS Path Planning Performance

The results presented in Section 6.7.2 provided insights into the accuracy of the localization system only. Uncertainty in SWS planning and control was deliberately removed from the evaluation. However, we were also interested in quantifying the performance of the planner itself. To do this, we examined the paths of the 10 trials to evaluate their consistency. This was done using the pose estimates provided by the localization module as ground truth. Since these estimates are assumed to be correct, localization uncertainty is removed from the analysis and the variance in path following can be attributed entirely to planner performance.

Fig. 6.15 shows three 20 meter sections of the route. The three sections were chosen arbitrarily except to evaluate locations with different densities of landmark features. Each section has ten paths shown (five in each direction), with the landmarks depicted as blue circles and the waypoints as black circles. Note the relative size of a landmark circle reflects its signature (its radius). Also note the radius of each waypoint (50 cm) includes the local planner's tolerance to reach the waypoint.

Qualitatively from the figure, planning performance is very good. To quantify this, we computed the 1D mean absolute error (MAE) and 1D standard deviation of the paths across a range of cross-sectional samples. These values were computed by taking 50 evenly spaced locations along the x axis. For each of these locations, the average normal was computed where it intersected the given x coordinate. Then the cross-sectional line was computed using the x coordinate and the mean y coordinate as a base point, and the average surface normal as the slope. The points of intersection of the paths along the cross-sectional line were then found and transformed to reduce the dimension to one via principal component analysis. The now 1D values were centered so that the mean value was zero. Finally, the data from each of the 50 cross sections were pooled to compute the MAE and standard deviation as it was in a compatible form. The three data sets were evaluated separately. The top figure had a MAE of 1.8 cm and a standard deviation of 2.1 cm. The middle figure had a MAE of 1.7 cm and a standard deviation of 2 cm. The bottom figure had a MAE of 1.9cm and a standard deviation of 2.4 cm.

When evaluated in conjunction with the localization performance reported in Section 6.7.2, we would expect the wheelchair to consistently drive the same path to a tolerance of approximately $\pm$ 10 cm. This is in fact consistent with our subjective observations made during testing.

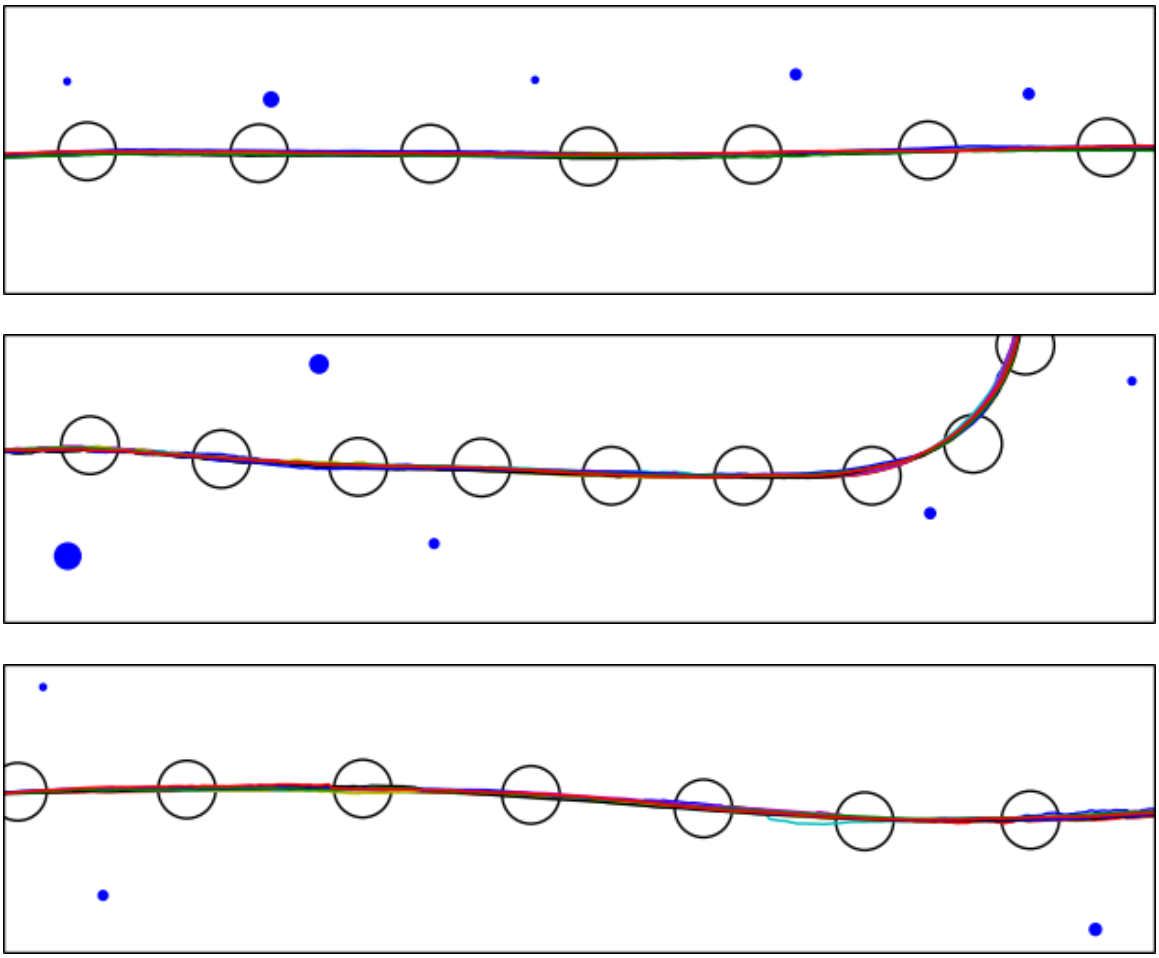

Figure 6.15: Close up of ten paths (five in each direction) on 20 meter sections of the block with various densities of landmarks. The landmarks are depicted in blue. The black circles indicate the tolerance for hitting a waypoint and have a radius of 50 cm.

6.7.5 Simulated User Testing

The results presented in Section 6.7.3 are limited in the sense that they do not reflect typical user operation. Specifically, it is the same route repeated 5 times in 2 directions. Ideally, user testing with non-confederate participants would be conducted. Due to concerns with human-subjects use and institutional review board (IRB) requirements, we chose to simulate such a use.

First, a total of 10 destinations of interest were identified in the test loop and a semantic label was attached to each. These are listed below. Note the numbers correspond to the locations highlighted in Fig. 6.9.

Path	Distance
$2 \rightarrow 5 \rightarrow 8$	572 m
$4 \rightarrow 10 \rightarrow 7$	943 m
$6 \rightarrow 5 \rightarrow 8$	385 m
$9 \rightarrow 10 \rightarrow 6$	463 m
$2 \rightarrow 1 \rightarrow 4$	199 m

Table 6.1: Results of simulated user trials, where each destination was randomly generated. All trials were successfully completed without incident.

1. Apartment 20 on Morton Street
2. Police station on Morton Street
3. Book store at Campus Square
4. Apartment 14 on Morton Street
5. Entrance to Whitaker Lab on Webster Street
6. Entrance to Whitaker Lab on Packard Avenue
7. Entrance to Mudd on Packard Avenue
8. Iaccoca plaza on Packard Avenue
9. Entrance to STEPS on Packard Avenue
10. Entrance to STEPS on Vine Street

We then performed five trials using random sampling without replacement. First, a random starting location was chosen. This was followed by a random intermediate goal location, and then a random final goal. Table 6.1 shows the path locations and total distance traveled for each trial. We should note that these trials were performed during sidewalk construction on Vine Street, so the link between destinations 1 and 10 had to be removed. As a consequence, some of the global plans (e.g, $4 \rightarrow 10 \rightarrow 7$) between locations were longer distance than usual.

All 5 trials were successfully completed without incident. These amounted to a total of 2.6 km of additional autonomous operations.

6.8 Discussion

In this chapter, we presented a system level approach to SWS navigation in urban environments. The proposed ecosystem features a mapping service which generates large-scale landmark maps, and the client SWS which integrates 3D perception for robust navigation in unstructured, outdoor environments. In our experiments, the SWS was able to reliably navigate over a distance of > 12 km without losing localization. From this, we claim the proposed map-based localization approach can provide the performance of a high-end GPS/INS system, but without the cost.

We are firm believers in the future of map-based localization solutions for urban environments that leverage 3D LIDAR/camera systems. We further note that since this work was completed, our 3D Hokuyo was rendered obsolete by the release of the Velodyne VLP-16 "Puck" LIDAR. In the same window, our IFM O3D200 3D cameras were also superseded by the release of the O3D303; the smaller, more accurate, lower-cost successor has 29 times the pixels of the O3D200. These higher performance sensors would only improve the performance of our current system.

There are of course concerns with map-based solutions to the localization problem. Two obvious questions are: 1) what happens if a feature is removed (e.g., a tree is cut down), and 2) what happens if a new feature is added (e.g., a tree is planted). Fortunately, the proposed approach is robust to such small-scale changes. In the former case, the removed feature will not be detected by the SWS and would be handled no differently than when a feature is occluded (i.e., no measurement update). In the latter case, the new feature *will* be detected by the SWS. However, there will be no respective feature in the landmark map. As a result, data association will likely fail so again the result will be a "nop." One could imagine a pathological case where a feature is removed, and a new featured added at approximately (but not exactly) the same location. If the new feature had the same signature as the old one, there is the potential for it to be wrongly associated in the landmark map. However, the error tolerance would have to be comparable to the uncertainty in wheelchair position which we have shown to be quite small. As a result, we expect the impact to localization accuracy in even this pathological case to be small.

Despite being relatively stable, large-scale changes *do* occur in urban areas. We saw this first-hand when the STEPS building was recently constructed at Lehigh University, affecting a roughly 100 meter stretch in our map. We also demonstrated in Section 6.7.3 that large-scale errors in the map (i.e., 15% false positive landmarks)

will result in localization failure. While there might be potential for SWS clients to update the map in crowd-sourcing fashion, in all likelihood remapping would be necessary in such an event.

While we were pleased with the navigation performance of the SWS, improvements are needed in the mapping service component. Specifically, false positives in landmark segmentation must be eliminated and an improved localization approach is necessary. However, in fairness these shortcomings are of less concern than if the issues were with the SWS itself. The Mapping Trike was fabricated from hardware on-hand, and within the constraints of our limited budget. In practice, such server vehicles would be viewed as part of the infrastructure. So, while the price of an SWS needs to be competitive for the consumer, cost constraints would be less of a concern for the mapping vehicle. As a result, we expect shortcomings in mapping performance could be readily addressed with improvements in hardware and/or software.

Although the SWS system described in this chapter was focused on outdoor sidewalk-level urban environments, the map based navigation approach should be generalizable to any large scale environment where natural features can be segmented and used as landmarks. To test this hypothesis, the next chapter describes an extension of the map-based localization approach to the domain of large scale indoor warehouse environments.

Chapter 7

Warehouse Mapping and Localization

In this chapter, we present a method for infrastructure-free localization of automated guided vehicles (AGVs) in a warehouse environment. This is an extension of the landmark map based localization approach described in the previous chapter (6), where a landmark map was used for localization in outdoor urban environments. The main difference between these approaches is the choice of natural features that are used as landmarks. Whereas the system in chapter 6 used sidewalk level pole-like features, the natural features used in the warehouse domain are the vertical pallet rack supports. The following sections describe the motivation for using this approach in warehouse environments, related warehouse localization work, the map generation procedure, the localization method, experimental results, and a brief discussion.

7.1 Background

In warehouse environments, manual forklifts (aka "lift trucks") are the workhorses of material handling [60]. In 2013 alone it was estimated that orders for over 1 million forklifts were placed [61]. Manual forklifts are driven by a human operator. A typical work cycle involves ferrying palletized goods and materials between storage (either racks and shelves or block storage areas) and trucks for receiving or shipment. Despite their flexibility and effectiveness in material handling tasks, they are not without their shortcomings. These include operating efficiency, high energy consumption, and safety considerations. These concerns have led to the rise of Automated Guided

Vehicle (AGV) systems for automated pallet transport and storage [62–65]. While they lack the adaptability of human operators, for repetitive material handling tasks they are far more efficient - often operating around the clock, and with reduced product damage. Also in large part due to rigorous ANSI safety standards regulating their design and use [66], they are far safer in practice.

AGVs are not merely vehicles, but autonomous *systems* in the full sense of the word. The components of an AGV system include one or more automated vehicles, a *localization system* which provides precise position and orientation estimates for the vehicles, a *route map* or network which delineates the guidepaths where the AGVs can travel, and a centralized controller which coordinates between and assigns specific tasks to the individual vehicles [67]. While all of these components are essential for AGV operation, it can be argued that the enabling technology is the localization system as it answers the first fundamental question which an autonomous vehicle must ask (i.e., Where am I?). There are multiple approaches to AGV localization [68]. The first systems relied upon either wire guidance, where wires are embedded in the floor and sensed inductively, or inertial guidance with magnets placed in the floor which act as landmarks to reset the dead reckoning system for overcoming drift. For contemporary AGV installations, the "gold standard" for localization is arguably laser guidance. In this paradigm, each AGV vehicle is fitted with a 2D scanning LIDAR system. As the AGV navigates, the LIDAR tracks precisely placed retro-reflector targets in the environment which serve as landmarks. Retro-reflectors are used so that the target's reflectivity can be used as a filter for robust landmark segmentation. Otherwise, attempting to segment targets in a 3D world using a 2D sensor is extremely error prone. In practice the approach works quite well, providing sub-centimeter positioning accuracy.

A significant drawback of each of the aforementioned approaches is that they require modifications to the warehouse infrastructure. In the case of laser-guidance, hundreds or even thousands of targets need to be accurately placed within the environment. The large number is necessary to ensure the required localization accuracy can be achieved from any location in the facility, as industrial environments are often very cluttered [67]. The result is not only significant per-vehicle costs, but also very high installation costs. Furthermore, the approach has limited flexibility in that changes to a facility's layout may necessitate significant modifications to the reflector installation.

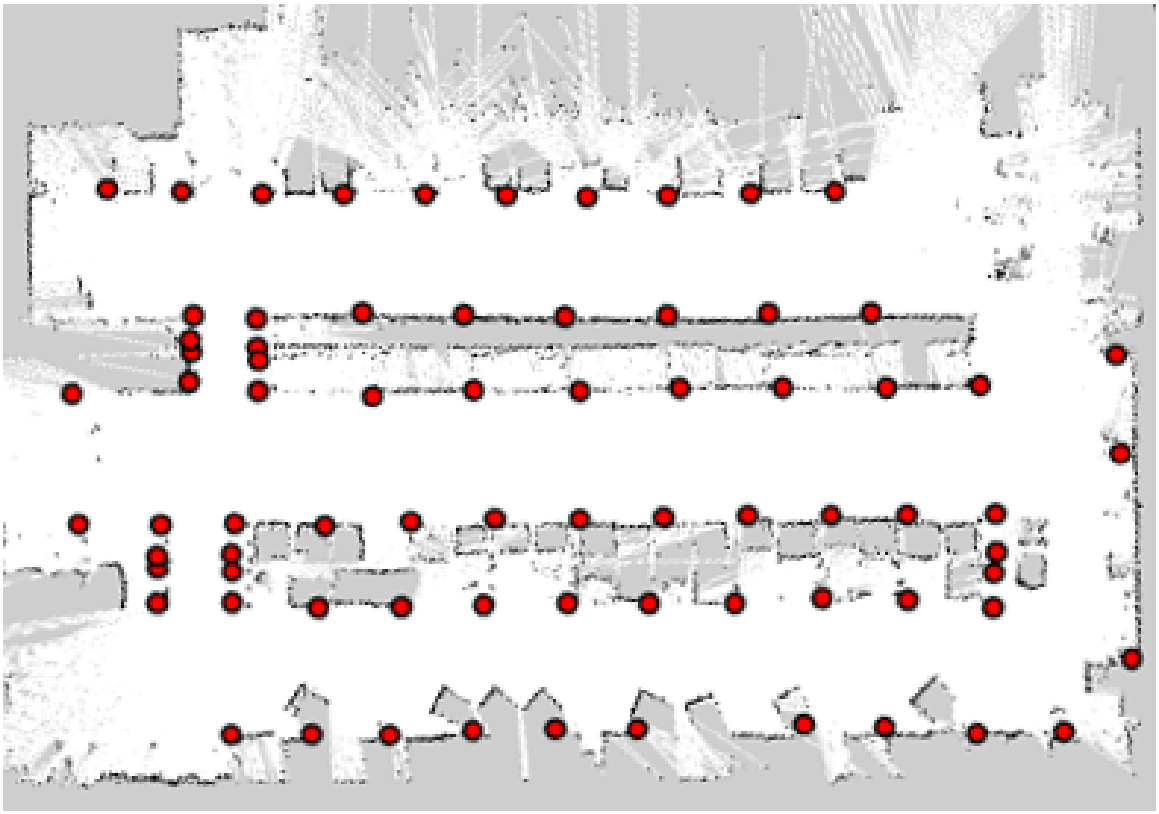

Figure 7.1: An occupancy grid map of an approximately 20 × 35 meter section of warehouse. The red circles denote a feature map composed of the vertical pallet rack supports.

It should then come as no surprise that there is significant interest in *infrastructure-free* solutions to AGV localization. Indeed, in their roadmap for boosting the use of AGVs in industrial applications, the authors in [67] specifically cite localization based upon natural landmarks as a key enabling technology. Both academic and industry researchers have taken notice. Vision-based solutions have been proposed in [68, 69], and most significantly by Seegrid Corporation [65]. These approaches come with the pros (e.g., lower-cost, passive sensors) and cons (e.g., limited robustness to illumination changes, inability to operate in darkness) associated with vision sensors. More recently, researchers have begun investigating the use of LIDAR systems to localize in large-scale warehouse environments by leveraging maps generated *a priori* [70, 71]. However, the authors' use of a 2D LIDAR for localization in a cluttered industrial environment necessitated a contour-based approach. Of concern is that palletized goods and materials may make up a significant portion of the LIDAR scan (contour), and the robustness of the approach to permutations in pallet placement and density

has only seen limited validation.

With these insights, we argue that:

1. Robust, infrastructure-free localization in a cluttered industrial environment motivates 3D LIDAR systems for perception

2. Feature based localization approaches are preferable, as they are more stable and less dependent upon variability in warehouse inventory, and

3. Using 3D LIDAR sensors, there are sufficient natural 3D features in warehouse environments that the use of retro-reflector targets can be eliminated.

Unfortunately, the accessibility of such an approach has been constrained due to the limited availability of suitable sensing technology. However this is rapidly changing, and AGV localization now appears ready for a paradigm shift.

Based on our previous work in landmark map-based localization in large scale urban environments [72], we propose an adaptation of the approach for large scale indoor warehouse environments. In urban environments, we used ubiquitous pole-like features, such as lamp posts, parking meters, and decorative trees, as natural landmarks in our map. Similarly, in warehouse environments, we propose using the pole-like pallet rack supports as landmarks. In addition to the new application area, the main extensions of this work include the integration of the Velodyne VLP-16 LIDAR, a comparison of alternate mapping paradigms, and methods used to segment the pallet rack supports from 3D representations of the warehouse environment.

7.2 Related Work

Mapping and localization in static environments is a mature area in robotics with a rich body of literature. As such, we limit the scope of discussion to the state-of-the-art in the specific context of warehouse environments.

In [73], the authors propose an infrastructure-free framework for warehouse navigation that uses a topological map rather than a globally consistent metric map. Their approach is to use a monocular camera to track the texture on the floor and the map is a locally consistent pose graph representation where each pose in the graph has an associated image. Localization is performed by matching the current camera frame to a likely subset of the pose graph. Due to the differences in map representation, our approach is not directly comparable.

More similar to our work in terms of map representation is a body of research [70, 71, 74] under the PAN-Robots project [75] funded by the European Union. The scope of the project is AGV based automation for factory logistics. The relevant sub-goals to our work are automated mapping and localization in warehouse environments. In [71] the authors use the GMapping algorithm [76] to build an 2D occupancy grid based map representation. We similarly use the GMapping algorithm, but only as a means to build a globally consistent 3D reconstruction of the environment. In [74], the authors extend the work to include GraphSLAM [26] based mapping approach where existing retro-reflectors were used as landmarks. This is similar to our approach in that the map representation is feature-based, however our landmarks are based on natural 3D features of the environment, rather than artificial 2D features. While a stated goal in [71] is to enable 3D mapping, all the mapping and localization work is 2D based. There is only a brief mention that 3D mapping is possible and is left as future work.

In [71, 74] they use a contour-based, adaptive Monte Carlo localization (AMCL) approach [26] where they use P-L-ICP [77] for visual odometry. In contrast, our localization approach focuses on naturally occurring warehouse features (the vertical uprights on the rack systems). We are able to do this by leveraging the latest in 3D LIDAR systems for localization, specifically the Velodyne VLP-16 LIDAR [78]. This enables us to robustly segment 3D features from the environment in real-time, and then associate these with 3D features in our landmark map. This 3D-to-3D feature mapping provides a significant advantage over 2D contour approaches in terms of system robustness. Furthermore, since the features we are tracking are temporally invariant for a given warehouse layout, we expect this approach will provide more stable and consistent localization performance.

7.3 Map Generation

The map representation used by the system is feature-based. In this work, we chose to use the vertical, pole-like supports of pallet racks as landmark features, which we will refer to simply as pole features. The mapping process consists of three main steps: (1) collect dense laser scans of the environment, (2) register the laser data to a common coordinate frame to create a 3D point cloud reconstruction of the environment, and (3) segment salient 3D features from the registered point cloud to create a landmark

map.

To acquire dense laser scans of the environment, we utilized our Mapping Trike platform 6.4.1. Details of the sensor suite are repeated here for convenience. An estimate of Mapping Trike's pose was provided by a Microstrain 3DM-GX3-45 inertial measurement unit (IMU) in conjunction with two 4096 cycles per revolution (CPR) resolution encoders mounted on the rear wheels. Two SICK LMS291-S14 LIDARs were mounted facing each side for the purpose of creating the 3D reconstruction. A single SICK LMS291-S05 LIDAR was mounted parallel to the ground plane facing the rear for the purpose of global pose corrections.

7.3.1 Coordinate Frame Registration

We note that although we are creating a 3D reconstruction of the warehouse environment and tracking 3D features, AGV localization will be on the plane. As a result, the mapping goal is register the 2D positions of the landmarks to a common global coordinate frame. There are many mature solutions to the Simultaneous Localization and Mapping (SLAM) problem for learning 2D maps. In this work, we chose to use the GMapping algorithm [76]. The main reason for this decision was convenience as a high quality open source solution implementation is available [79]. However, any SLAM algorithm capable of recovering the trike's trajectory could be a suitable replacement.

The GMapping algorithm is a particle filter based approach to learn a 2D occupancy grid representation of the environment from horizontal planar laser scans and odometry measurements. Laser scans from the rear-facing LIDAR were used as input. For the odometry measurements, an extended Kalman filter (EKF) was used which incorporated the vehicle kinematics in the predictive step and data from the encoder and IMU for the corrective step. Since the floor of a warehouse environment is roughly planar, we used a 3 degree of freedom motion model of the form:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k+1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k} + \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta_l \\ \Delta_a \end{pmatrix},$$

where $\mathbf{x}$ denotes the state vector consisting of the Cartesian coordinates (x, y) and the yaw angle θ, Δ_l and Δ_a were the linear and angular displacements over some small discrete time step denoted as k. Note that there is an unmodeled kinematic constraint from the trike's front wheel, but this was not needed for mapping purposes.

The GMapping algorithm learns a 2D occupancy grid representation of the environment. However, our goal is to learn a 2D feature-based representation by creating a 3D reconstruction of the environment, segmenting the landmarks from this representation, and then registering the landmarks to the 2D plane. We used a structure-from-motion based approach to register the side-facing LIDAR scans to the GMapping result, which only required logging the pose corrections at each time step of the GMapping algorithm.

7.3.2 Landmark Segmentation

With the corrected trike poses, a globally consistent 3D point cloud reconstruction of the environment could be constructed as depicted in Fig. 7.2 (center). The goal was to segment the pole features from this reconstruction. First each 2D laser scan was transformed to a coordinate frame where the x axis was parallel to the ground plane, here denoted as a sequence of consecutive points, $S = (p_1, \ldots, p_k)$, where k is the number of points in a scan and each p_i is a vector of coordinates, $[x, y]^T$.

Then each scan was clustered into subsets where the points in a valid cluster satisfied three conditions:

1. $\|p_{i+1} - p_i\| < \alpha$, where α is the distance tolerance between points,
2. $\|a^T p_i - f(S)\| < \beta$, where $a = [0, 1]^T$, $f(S)$ is the median y value of the points in scan S and β is a distance tolerance to $f(S)$,
3. $\|p_{n_c} - p_{1_c}\| > \gamma$, where p_{1_c} and p_{n_c} are the first and last points in cluster c respectively, and γ is a tolerance on the length of a cluster.

The assumption was that a pallet rack support was the tallest contiguous vertical object in a scan that contained one. Hence, the first condition validates "contiguousness", the second condition validates "verticality", since the median value should lie on the pole feature, and the third condition validates "tallness". In this work, the α and β values were both set to 2 cm, and γ was set to 30 cm since the side-facing LIDARs were not perfectly perpendicular to the ground.

The points from all valid clusters from all laser scans were transformed to the 3D global coordinate frame based on the corrected trike poses and combined into a set M. The points in M were clustered into subsets that satisfied the condition $\|p_i - p_j\| < \epsilon$ where ϵ is the maximal intra-cluster distance between points. In this work, ϵ was

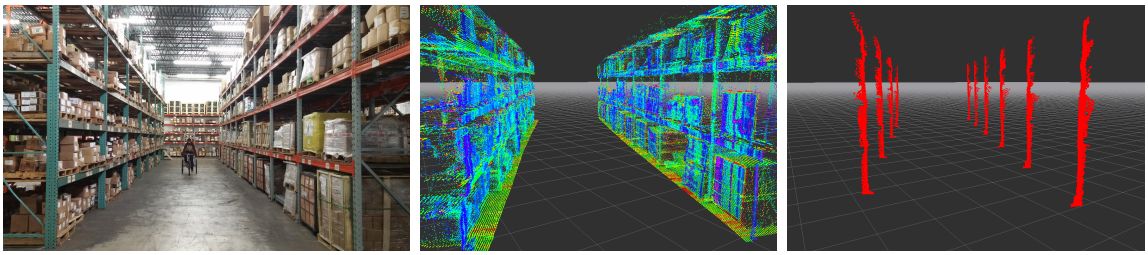

Figure 7.2: (Left) One aisle of the warehouse. (Center) A point cloud reconstruction. (Right) Landmark segmentation for feature map generation.

set to 5 cm. Any cluster that had a bounding box height greater than 2 meters was considered to be a landmark. A depiction of accepted landmark clusters is shown in Fig. 7.2 (right). The 2D landmark map was then synthesized by projecting centroids of each accepted cluster to the ground plane.

7.4 Map-based Localization

This section describes the proposed AGV localization procedure, which assumes that a landmark map is available. The main components of the system are perception and localization. The perception component handles landmark segmentation and the localization component maintains an estimate of the AGV's pose within the landmark map. A description of each of these components and the development platform now follows.

7.4.1 The Development Platform

Since an AGV was not initially available to us, to demonstrate feasibility we employed our Smart Wheelchair (SWS) platform as a surrogate 6.6. The SWS was equipped with a mast-mounted Velodyne VLP-16 for exteroceptive sensing. The VLP-16 was chosen as it provides accurate, real-time 360° 3D measurements. It features 16 laser sensors with a vertical field of view of 30°. We also note that the VLP-16 costs less ($\approx$ \$8000 USD) than single-beam LIDARs currently used on AGV reflector-based localization systems. The SWS also provided odometry information from wheel encoders and IMU measurements.

7.4.2 Landmark Segmentation

We denote a 360° VLP-16 laser scan as a matrix of the form:

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \dots & p_{1,k} \\ \vdots & \vdots & \dots & \vdots \\ p_{16,1} & p_{16,2} & \dots & p_{16,k} \end{pmatrix},$$

where k is the number of scans, $p_{i,j}$ is a 3D coordinate. Each row corresponds to a laser scan and each column corresponds to an azimuth angle.

The first step is to correct the measurements in P to compensate for vehicular motion. We assume constant motion between odometry measurements and a constant rotational velocity of the sensor. For each column j in P, the corrected translation is given by $lerp(w_0, w_1, \eta)$ where $lerp(\cdot)$ is the linear interpolation operation, w_0 and w_1 are the measured vehicle position at the beginning and end of the scan respectively, and $\eta = (j-1)/(k-1)$ is the interpolation parameter. Similarly, the corrected rotation for each column j in P is given by $slerp(q_0, q_1, \eta)$, where $slerp(\cdot)$ is the spherical linear interpolation operation and q_0 and q_1 are quaternion representations of the measured vehicle yaw at the beginning and end of the scan respectively.

Next, the points in each column j in P are associated to 2D polar coordinates, $(\rho_{i,j}, \phi_{i,j})$, based on the x and y components of each point, assuming that the $x-y$ plane is the ground. Then a set of valid points was created:

$$V = \{p_{i,j} \mid \|\rho_{1,j} - h(j)\| < \delta \wedge \ldots \wedge \|\rho_{16,j} - h(j)\| < \delta\},$$

where $h(\cdot)$ is the median ρ value in column j and δ is a distance tolerance from this value. In this work the value of δ was set to 3 cm. The assumption was that due to the limited vertical field of view and *a priori* knowledge of the sensor position that all 16 scans at a given azimuth would hit a pole feature.

The points in V were then clustered into subsets that satisfied the condition $\|\pi_{xy}(p_i) - \pi_{xy}(p_j)\| < \xi$ where $\pi_{xu}(p)$ is the $x-y$ projection of point p and ξ is the maximal intra-cluster distance between points. In this work, ξ was set to 3 cm. Each cluster was then validated based on the geometry of its oriented bounding box. Clusters were rejected if: the height was less than 1.2 meters, the max(width, depth) was greater than 0.5 meters, or the ratio of the height to the max(width, depth) was less than 5. Fig. 7.3 depicts the resulting segmentation. The segmentation operated in real-time with a VLP-16 scan rate of 5 Hz.

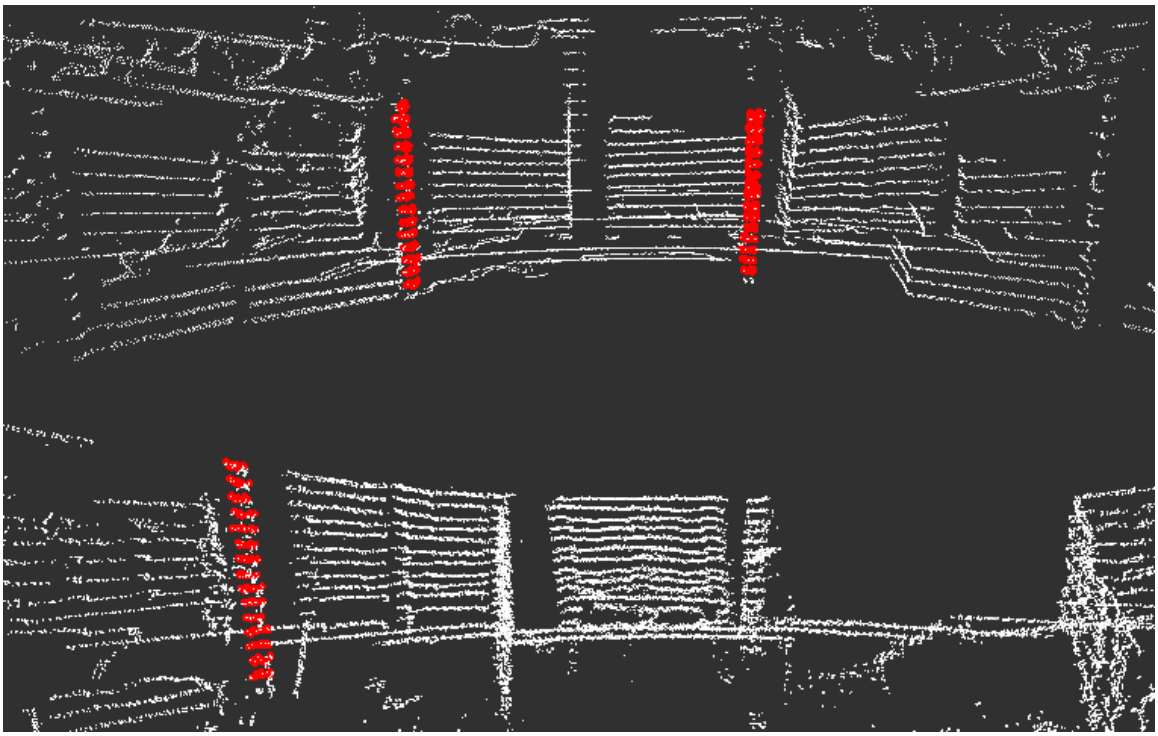

Figure 7.3: Real-time landmark segmentation for localization using the Velodyne VLP-16. Detected landmarks are highlighted in red.

7.4.3 Localization Procedure

Localization of the SWS was performed using the same particle filter based approach described in our previous work [72]. In short, we implemented a variant of the feature-based FastSLAM 2.0 algorithm [25] that performed localization, but no mapping. We found that due to the improved proposal distribution, fewer particles were needed to maintain localization. In this work, we used 20 particles for our localization experiments. This compares favorably to the AMCL approach in [71] which required hundreds of particles for effective localization. For pose estimation the mean over the particle set was used.

7.5 Experiments

To demonstrate the effectiveness of the proposed method in its intended environment, a map was constructed of an approximately 20 $\times$ 35 meter section of warehouse shown in Fig. 7.1. The following sections describe the mapping results and the client localization results. We leveraged the Robot Operating System (ROS) [27] framework

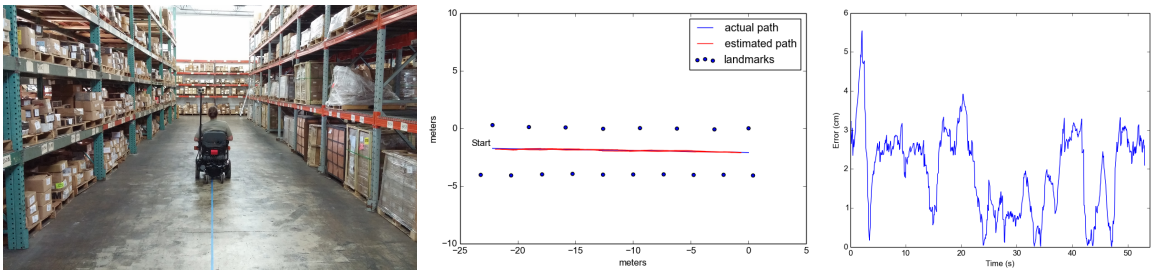

Figure 7.4: (Left) The localization accuracy experiment set up with the SWS equipped with a Velodyne VLP16. (Center) The actual vs. estimated path. (Right) Absolute error vs. time. The mean absolute error was 1.9 cm.

for our implementation and also utilized the Point Cloud Library (PCL) [59] for processing point cloud data from the exterioceptive sensors.

7.5.1 Mapping

To construct the map, data were collected by driving the Mapping Trike through the warehouse at a rate of approximately 1 m/s. In conjunction with the LMS291 LIDAR scan rate of 75 Hz, this gave us a vertical scan for each 1-2 centimeters traveled which was sufficiently dense for our landmark segmentation procedure. An advantage of the Mapping Trike platform in this scenario was that the maneuverability allowed us to collect data without disrupting normal warehouse operations.

To validate the effectiveness of the landmark segmentation of the Mapping Trike, the number of landmarks was counted by hand as a ground truth measure. This value was compared against the map generated by the mapping process. In total, 74 of 74 visible landmarks were successfully segmented. Furthermore, no false positives were detected. From this, we report that in our test environment the proposed mapping approach had a 100% true positive rate for segmenting landmarks, and a 0% false positive rate. While there were no ground truth measurements of the true landmark coordinates to determine a quantitative measure of accuracy, the generated map was qualitatively consistent to the environment based on the spacing between landmarks and the ability of the SWS to localize.

A second mapping experiment was performed to evaluate the viability of mapping with the SWS platform. The motivation for this was to investigate the potential for eliminating the special purpose mapping vehicle. In other words, could an AGV equipped with a VLP-16 LIDAR create its own map? While the accuracy of the

VLP-16 in conjunction with the motion estimation error of the platform was initially deemed unsuitable to create an accurate 3D reconstruction, the SWS (our AGV surrogate) had the advantage of being able to detect each landmark multiple times from different poses. Thus, a given landmark position estimate could be derived from multiple constraints.

To build the map, data were logged while driving several loops through the warehouse (the same data set used in Fig. 7.5), landmarks were segmented using the approach in Section 7.4.2, and then a classic EKF SLAM formulation [80] was used to learn the map. The SWS generated map contained all the ground truth landmarks with no false positives.

To quantitatively compare the maps a minimization of the form:

$$\underset{R,t}{\operatorname{argmin}} f(R,t) = \frac{1}{N}\sum_{i=1}^{N} \|x_i - Ry_i - t\|^2,$$

where x_i and y_i are the corresponding points, N is the number of points, R is a rotation matrix in $SO(2)$, and t is a translation vector, was solved to obtain the obtain the R and t to align the points. After the points were aligned the distances between corresponding points were computed and used as a quantitative measure of map similarity. This resulted in a mean distance of 4.3 cm with a standard deviation of 3.5 cm. Again, while we lack absolute ground truth, the relative consistency of the two maps indicates the potential for an AGV equipped with a Velodyne VLP-16 LIDAR to create its own landmark map.

7.5.2 Localization Accuracy

In an attempt to quantify localization accuracy, we performed an experiment where we marked a line down the center of the middle aisle with blue tape, shown in Fig. 7.4 (left). The line served as "ground truth" and was measured by hand relative to the landmark positions. The operator then attempted to drive the SWS straight down the line, turn in place at the end, and drive straight back to the starting position. Results of localization are shown in Fig. 7.4. The center figure depicts the path traveled and the right figure depicts the absolute error to the target line over time. The average path error was 1.9 cm with a standard deviation of 1.1 cm and a median error of 2.3 cm. While this is in excess of our target centimeter-level accuracy, we note that the analysis assumes that the line was perfectly straight (it was not) and

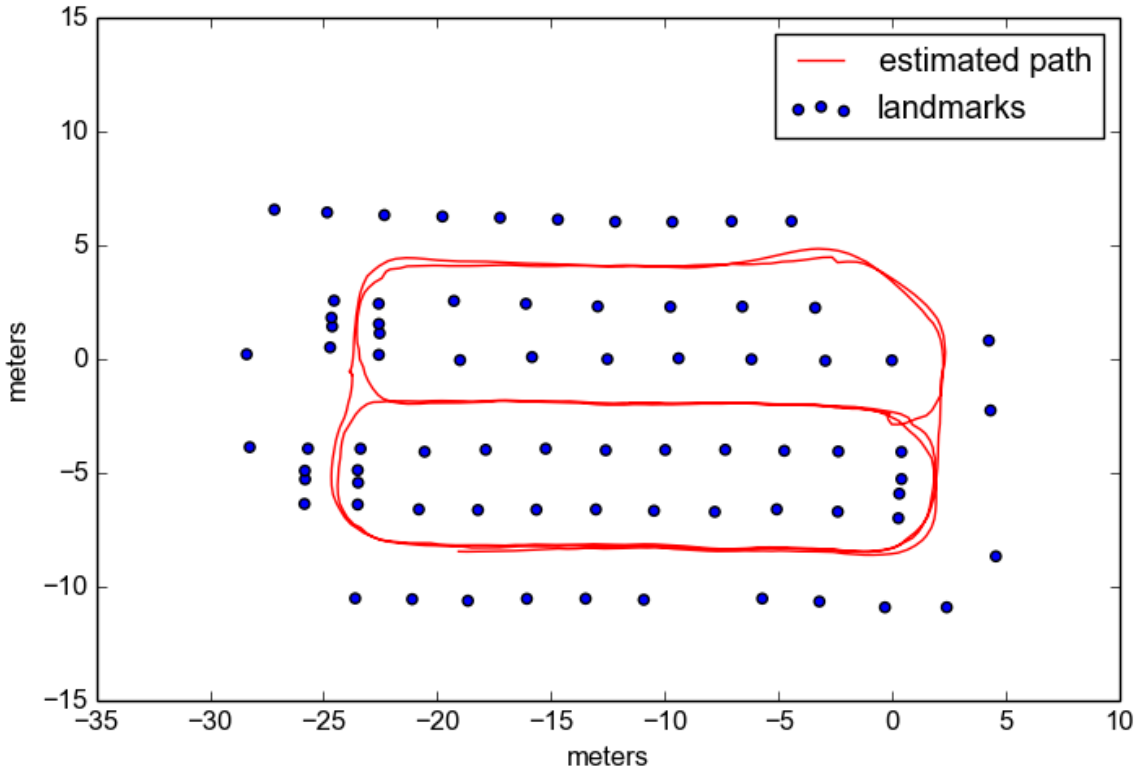

Figure 7.5: The estimated real-time corrected path where the SWS was driven around several arbitrary loops within the warehouse map.

the SWS operator drove down the center of the line (he did not). This is evidenced in Fig. 7.4 (right), where there is a noticeable symmetry about the 30 second mark when the turn in place maneuver was performed. This is likely due to a bias error in the placement of the tape line. As a result, centimeter-level localization accuracy may have been achieved. We acknowledge that this was a limited test, but argue that it demonstrates the potential of our approach to achieve the necessary localization accuracy for an infrastructure-free warehouse environment.

To investigate the ability to maintain localization over time, a second experiment was performed where the SWS was driven around several arbitrary loops in the map at an average speed of 1.0 m/s for approximately six minutes. The estimated path is shown in Fig. 7.5. The starting pose was approximately $[-19\ -9\ 0]^T$, and the total distance traveled was 353 meters. There were an average 9.5 landmark observations per meter traveled. Every visible landmark was detected, and there were no false positive (a feature being incorrectly associated with a landmark) as accurate

localization was maintained throughout the trial. A video showing the 3D reconstruction and landmark segmentation for part of this experiment can be viewed at `https://www.youtube.com/watch?v=B73tgGoT6Ms&feature=youtu.be`.

7.6 Discussion

In this chapter, we presented a 3D map-based approach to infrastructure-free localization of an AGV in a warehouse environment. Preliminary results support our hypothesis that there are sufficient natural 3D landmarks to support a robust, feature-based localization approach. During warehouse mapping, visible pole features were segmented with 100% reliability, and with a 0% false positive rate. During localization trials, all visible landmarks were detected, and again no false positive landmarks were identified. The demonstrated accuracy of the localization system was approximately 2cm.

This research was an extension of our outdoor mapping and localization approach described in the previous chapter 6. One significant improvement that we observed herein was the potential for eliminating the special purpose mapping vehicle previously required. This is a side-effect of employing the recently introduced Velodyne VLP-16 LIDAR on our AGV surrogate. In chapter 6, we relied upon an actuated Hokuyo UTM-30LX for 3D perception. Compared to the 3D Hokuyo, the VLP-16 has over 3X the range, 2X the number of effective beams, up to 2.5X the angular resolution, and greater fields-of-view. Our experimental results indicate that these improvements are sufficient to enable mapping with the AGV platform itself, which dramatically simplifies the logistics for a real-world implementation.

Chapter 8

Discussion & Future Work

This dissertation presented three autonomous navigation tasks relying on 3D point cloud based object detection. The specific objects in each case were chosen such that they were semantically interesting for the task at hand. The enabling technology for these tasks was affordable 3D sensors and the computational power to perform object detection in real time. These factors also influenced algorithmic decisions in the navigation components: map representation, localization and path planning. All of these systems had to operate in real-time sharing the same computational resources.

A key decision for our autonomous navigation applications was the choice of map representation. The map representation is the foundation of robot navigation; localization and path planning algorithms depend on the map representation. In turn, these algorithms are fundamentally coupled to the capabilities of the sensors involved and the perceptual algorithms used. Much of the work in this dissertation has focused on the real-time 3D point cloud based object detection systems, but similar navigation components were used throughout the three systems described in chapters 5, 6, and 7. The following sections discuss some extensions related to the interplay between the 3D perception and aspects of environmental modeling and navigation not explored in this dissertation.

8.1 Richer Semantic Map Representations

There are three main interrelated map representations used in this dissertation: a globally consistent landmark feature map, a route network to demarcate safe paths relative to the landmark map, and a local occupancy map for path planning purposes.

The main design decisions were based on integrating the point cloud based object detection with the navigation systems. For instance, although we use 3D information for object detection, these objects are reduced to 2D representations for efficient localization and path planning algorithms. The experiments in this dissertation never used the full computational resources available and we believe we can make further use of sensor information. This section discusses augmentations to initial map representation to encode additional semantic information into the map with the goal of enhancing navigation performance.

In chapter 6 a route network was used for global planning purposes to demarcate a safe path relative to the landmark map. However, the route network does not express margins of safety along the path (e. g. the extent of the sidewalk). The local planner is not necessarily constrained to remain on the path to circumvent obstacles and may plan a trajectory that is outside the bounds of the sidewalk. We have done preliminary work in terrain classification to automatically distinguish between sidewalk, asphalt, grass, and "other" terrain classes using remission measurements from the O3D200 3D cameras [81]. It would be straightforward to embed this additional semantic information into the local occupancy grid by adding one additional value per cell. The sample based planning algorithm could make use of this additional information by adding an extra term to the cost function. We expect this additional terrain information information will improve navigation performance over geometric information alone.

Furthermore, the landmark map representation encoded only the 2D position of a landmark and the radius of the feature. The radius was crucial for reducing false data associations during localization. A natural extension for this is to encode more semantic information for each landmark derived from the detected object's point cloud to further aid in data association. A further extension is to use more categories of objects for landmarks or have more specificity within a category. For example, the pole-like features in the urban environment could be recognized as belonging to a specific class, such as tree or lamp post, rather than the more general pole-like object with a given radius. Another factor not explored here is the incorporation of additional sensing modalities to aid in object recognition. The Kinect based sensors can register a color to point in the point cloud. There has been some work in the area of semantic object recognition using both color and depth [32, 33, 82, 83]. Similar methods could be used to obtain additional semantic information for each landmark.

8.2 Modeling Environment Dynamics

Traditionally, learning the map representation is done by solving a Simultaneous Localization and Mapping (SLAM) problem on sensor data gathered from a single pass through the environment. This map is then used for future localization and path planning tasks. And this is the approach taken in chapters 6 and 7. The problem with this approach is that for long term autonomy, the assumption that the environment is stationary is not necessarily valid. As long term navigation in dynamic environments is one of the main goals of this work, this section discusses approaches to continuously model the environment in order to adapt to change.

An approach to this is to classify aspects of the environment by the degree of temporal stability they have: stationary, temporarily stationary, or dynamic. For example, given an outdoor urban environment, buildings could be considered stationary, parked cars as temporarily stationary, and pedestrians as dynamic. In our work, we implicitly modeled low dynamic obstacles by choosing landmarks that were assumed to be temporally stable or stationary, or that the long term existence of the landmark did not matter, such as in the case of CoPilot. However, on long time scales this assumption is not true. Take the warehouse domain for instance, the pallet racks may change configuration at some point. We want to allow for the detection and correction of the map if possible in these scenarios. The following sections discuss modeling low dynamic objects and high dynamic objects respectively.

8.2.1 Low Dynamics and Map Maintenance

As mentioned above, we chose the features in the environment that were assumed to have a high level of temporal stability. In the discussion in chapter 6 we also mentioned that large scale changes have occurred in our outdoor urban map as a result of construction, which necessitated a remapping of certain sections of the map. An interesting open question is can changes in landmark based maps be detected and recovered from?

There is some research in modeling low dynamics, but in the context of visual maps. Due to the dense information from a visual sensor, distinguishing a particular scene can potentially be done with a single frame. Because of this property, visual mapping methods typically represent the map as a graph of poses where each node

represents a 6 degree of freedom pose with an associated image frame from the camera and edges represent relative geometric constraints between nodes. Localization methods match the current image frame against the nodes in the pose graph to derive the robot's pose. This is called place recognition. An interesting property of visual maps is that they can encode conflicting information, for example two images of a door, one open and one closed, can exist in a visual map.

A useful concept proposed in [84] is the idea of a lifelong visual map. A lifelong map should be able to handle the following situations: incremental mapping, dynamic environments, and localization and odometry failure. Incremental mapping is the ability of the system to add new sections to the map at any point, that is the system is continuously localizing and mapping. With respect to dynamic environments, the system should be able to repair its map to reflect changes in the environment. The system should recover from localization failure by relocalizing in the map at the earliest opportunity.

An additional method proposed in [85] is to encode the map based on visual experiences where each experience is a sequence of images with relative metric information about the robot's path. Places that change over time are represented as a set of experiences, that is, the number of experiences associated with a specific place is proportional to its dynamics. Localization is performed by data association (place recognition) with all previous experiences and when localization fails, a new experience is recorded.

The key system that enables both lifelong visual maps and visual experience based maps is place recognition. Place recognition in landmark maps is challenging because the landmarks are typically sparse point features and constellations of these features are ambiguous from different vantage points [86]. The ability to disambiguate constellations of landmarks can be achieved by encoding additional semantic information into the landmark representations, as mentioned in the previous section, and using this information to help solve the correspondence problem. The pursuit of this line research can enable landmark maps to achieve the benefits of lifelong visual maps and visual experience based maps. The ability to maintain multiple representations of the same place can aid in more robust localization and the potentially the ability to detect and recover from localization failures.

The disadvantage to these long term map representations is the unbounded growth in the map when attempting to maintain the perceptual history of a given place. The

work in [87] tackles the unbounded growth in visual maps when accumulating map data over an unbounded number of sessions. Their approach is based on online localization and offline map maintenance. The main idea is to generate compact maps offline from the subset of all the recorded data that minimize map size and maximize localization usefulness online. These summary maps are downloaded by the vehicle before a mission. Although their method is based on visual maps, this concept resonates with certain aspects of our work. In chapter 6 the map representation that the client vehicle uses is downloaded from a cloud based component. Also, in chapter 7 the viability of using the client vehicle to learn the map. A system that combines these ideas could have a central cloud based map service where map data is constantly collected as the client vehicles perform their navigation tasks. The cloud service could offer the best current map when a client vehicle is about to embark on a new mission. A system such as this could potentially detect environmental changes that should be reflected in the landmark map automatically.

8.2.2 Tracking High Dynamic Obstacles

In some contemporary literature, handling dynamic obstacles has been done by unifying the environmental dynamics into a single representation. For example, in [88] moving objects are modeled using an occupancy grid under the assumption that that occupancy is caused by objects and when objects move, the corresponding occupied cells of the map should move accordingly. Their Bayesian occupancy filter representation tracks the dynamics of every grid cell over time. Similarly, in [89] an occupancy grid based approach where the occupancy of each cell is modeled as a hidden Markov model. Instead of using a single representation, some approaches separate the dynamic and static aspects of the environment by maintaining two maps. For example, in [90] a general framework is devised to simultaneously localize, map, and track dynamic objects by determining whether a measurement is caused by a static or dynamic object and placed in an appropriate map accordingly.

These approaches are similar in that occupancy grid maps are used to model the motion of dynamic obstacles. Our existing path planning system only models the instantaneous occupancy of dynamic obstacles in the local costmap and plans accordingly. Since the dynamics of moving obstacles are not modeled, there exists the possibility of collision. By adding additional information to each grid cell that encodes motion, the path planner could make use of this information for more reliable

collision avoidance. This information could be derived from an 3D point cloud based object detection methods for specific objects known to be dynamic.

For instance, one significant aspect of urban environments that was ignored in this work was navigation in crowds. This is a rich research area in its own right, and one we intend to investigate in the future. One insight we are happy to report is that on the whole, pedestrians are largely considerate of the SWS and give it a wide birth. We discovered through many hours of testing that despite its sensor "warts," the SWS is perceived merely as a person operating a conventional EPW and not as a smart wheelchair system. We believe this "disguise" will be an asset when navigating crowded environments.

An area for future research is to first focus on the detection and tracking of pedestrians under the assumption that pedestrians are the most common class of dynamic obstacle in environments where EPWs operate. This can be achieved by finding a way to segment pedestrians from point cloud data. The basic aim would be to devise an online detection and tracking method that can be encoded into the local costmap to assist in path planning. A further extension could make use of tracked pedestrian data to model pedestrian interaction with wheelchairs which could used to model the behavior of crowds. This extension could enable the path planner to move with the flow of foot traffic.

Bibliography

[1] Thrun, S. *et al.* Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium* **1**, 1–35 (2002).

[2] DeSouza, G. N. & Kak, A. C. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence* **24**, 237–267 (2002).

[3] Kostavelis, I. & Gasteratos, A. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems* **66**, 86 – 103 (2015). URL `http://www.sciencedirect.com/science/article/pii/S0921889014003030`.

[4] Guizzo, E. How googles self-driving car works. *IEEE Spectrum Online, October* **18** (2011).

[5] Urmson, C. *et al.* Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics* **25**, 425–466 (2008).

[6] Montemerlo, M. *et al.* Junior: The stanford entry in the urban challenge. *Journal of field Robotics* **25**, 569–597 (2008).

[7] Bacha, A. *et al.* Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics* **25**, 467–492 (2008).

[8] Leonard, J. *et al.* A perception-driven autonomous urban vehicle. *Journal of Field Robotics* **25**, 727–774 (2008).

[9] Bohren, J. *et al.* Little ben: the ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics* **25**, 598–614 (2008).

[10] Miller, I. *et al.* Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics* **25**, 493–527 (2008).

[11] Nguyen, A. & Le, B. 3d point cloud segmentation: A survey. In *Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on*, 225–230 (IEEE, 2013).

[12] Douillard, B. *et al.* On the segmentation of 3d lidar point clouds. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2798–2805 (IEEE, 2011).

[13] Teichman, A., Levinson, J. & Thrun, S. Towards 3d object recognition via classification of arbitrary object tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 4034–4041 (IEEE, 2011).

[14] Teichman, A. & Thrun, S. Tracking-based semi-supervised learning. *The International Journal of Robotics Research* **31**, 804–818 (2012).

[15] Endres, F., Plagemann, C., Stachniss, C. & Burgard, W. Unsupervised discovery of object classes from range data using latent dirichlet allocation. In *Robotics: Science and Systems*, vol. 2, 113120 (Seattle, Washington;, 2009).

[16] Anguelov, D. *et al.* Discriminative learning of markov random fields for segmentation of 3d scan data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 169–176 (IEEE, 2005).

[17] Triebel, R., Shin, J., Siegwart, R., Siegwart, R. Y. & Siegwart, R. Y. Segmentation and unsupervised part-based discovery of repetitive objects. In *Robotics: Science and Systems*, vol. 2 (2010).

[18] Spinello, L., Arras, K. O., Triebel, R. & Siegwart, R. A layered approach to people detection in 3d range data. In *AAAI* (2010).

[19] Wang, D. Z., Posner, I. & Newman, P. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 4038–4044 (IEEE, 2012).

[20] Szeliski, R. *Computer vision: algorithms and applications* (Springer Science & Business Media, 2010).

[21] Balasubramanian, R., Das, S., Udayabaskaran, S. & Swaminathan, K. Quantization error in stereo imaging systems. *International journal of computer mathematics* **79**, 671–691 (2002).

[22] Teichman, A., Miller, S. & Thrun, S. Unsupervised intrinsic calibration of depth sensors via SLAM. In *Robotics: Science and Systems* (2013).

[23] Fischler, M. & Bolles, R. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Communications of the ACM* (1981).

[24] Gao, C. & Spletzer, J. R. On-line calibration of multiple lidars on a mobile vehicle platform. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 279–284 (IEEE, 2010).

[25] Montemerlo, M. & Thrun, S. Fastslam 2.0. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics* 63–90 (2007).

[26] Thrun, S., Burgard, W. & Fox, D. *Probabilistic robotics* (MIT press, 2005).

[27] Quigley, M. *et al.* Ros: an open-source robot operating system. In *ICRA workshop on open source software*, vol. 3, 5 (2009).

[28] LaValle, S. M. *Planning algorithms* (Cambridge university press, 2006).

[29] U.S. Department of Health and Human Services, Administration on Aging. A profile of older americans: 2011 (2011).

[30] Simpson, R., LoPresti, E. & Cooper, R. How many people would benefit from a smart wheelchair? *Journal of Rehabilitation Research & Development* **45**, 53–72 (2008).

[31] Rusu, R. B., Meeussen, W., Chitta, S. & Beetz, M. Laser-based perception for door and handle identification. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 1–8 (IEEE, 2009).

[32] Gupta, S., Arbelaez, P. & Malik, J. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 564–571 (IEEE, 2013).

[33] Ren, X., Bo, L. & Fox, D. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2759–2766 (IEEE, 2012).

[34] Levine, S. P. *et al.* The navchair assistive wheelchair navigation system. *Rehabilitation Engineering, IEEE Transactions on* **7**, 443–451 (1999).

[35] Yanco, H. A. Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. In *Assistive technology and artificial intelligence*, 256–268 (Springer, 1998).

[36] Parikh, S. P. *et al.* Human robot interaction and usability studies for a smart wheelchair. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4, 3206–3211 (IEEE, 2003).

[37] Derry, M. & Argall, B. Automated doorway detection for assistive shared-control wheelchairs. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1254–1259 (IEEE, 2013).

[38] Harris, C. & Stephens, M. A combined corner and edge detector. In *Alvey vision conference*, vol. 15, 50 (Manchester, UK, 1988).

[39] American's with disability act standards for accessible design. `http://www.ada.gov/2010ADAstandards` (2010).

[40] Simpson, R. C. Smart wheelchairs: A literature review. *Journal of rehabilitation research and development* **42**, 423 (2005).

[41] Perry, D. Medical never-never land: Ten reasons why america is not ready for the coming age boom.(p. 1). *Alliance for Aging Research* (2002).

[42] Irie, K. & Tomono, M. Localization and road boundary recognition in urban environments using digital street maps. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 4493–4499 (IEEE, 2012).

[43] Yokozuka, M., Suzuki, Y., Hashimoto, N. & Matsumoto, O. Robotic wheelchair with autonomous traveling capability for transportation assistance in an urban environment. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2234–2241 (IEEE, 2012).

[44] Simonite, T. Data shows Googles robot cars are smoother, safer drivers than you or I. *Technologyreview, Oct* **25** (2013).

[45] United Nations Department of Economic and Social Affairs. *World Urbanization Prospects: The 2014 Revision* (United Nations Publications, 2015).

[46] Gao, C., Sands, M. & Spletzer, J. R. Towards autonomous wheelchair systems in urban environments. In *Field and Service Robotics*, 13–23 (Springer, 2010).

[47] Gao, C., Miller, T., Spletzer, J. R., Hoffman, I. & Panzarella, T. Autonomous docking of a smart wheelchair for the automated transport and retrieval system (atrs). *Journal of Field Robotics* **25**, 203–222 (2008).

[48] Gao, C., Hoffman, I., Panzarella, T. & Spletzer, J. Automated transport and retrieval system (atrs): A technology solution to automobility for wheelchair users. In *6th Conference on Field and Service Robotics (FSR07)* (2007).

[49] Montella, C., Perkins, T., Spletzer, J. & Sands, M. To the bookstore! autonomous wheelchair navigation in an urban environment. In *Field and Service Robotics*, 249–263 (Springer, 2014).

[50] Doshi, F. & Roy, N. Efficient model learning for dialog management. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, 65–72 (IEEE, 2007).

[51] Bostelman, R. & Albus, J. Sensor experiments to facilitate robot use in assistive environments. In *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, 8 (ACM, 2008).

[52] Cooper, R. *et al.* Personal mobility and manipulation appliancedesign, development, and initial testing. *Proceedings of the IEEE* **100**, 2505–2511 (2012).

[53] Anguelov, D. *et al.* Google street view: Capturing the world at street level. *Computer* 32–38 (2010).

[54] Chong, Z. *et al.* Synthetic 2d lidar for precise vehicle localization in 3d urban environment. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1554–1559 (IEEE, 2013).

[55] Baldwin, I. & Newman, P. Road vehicle localization with 2d push-broom lidar and 3d priors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2611–2617 (IEEE, 2012).

[56] Lategahn, H., Schreiber, M., Ziegler, J. & Stiller, C. Urban localization with camera and inertial measurement unit. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, 719–724 (IEEE, 2013).

[57] Kummerle, R., Ruhnke, M., Steder, B., Stachniss, C. & Burgard, W. A navigation system for robots operating in crowded urban environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 3225–3232 (IEEE, 2013).

[58] Fischler, M. A. & Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* **24**, 381–395 (1981).

[59] Rusu, R. B. & Cousins, S. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 1–4 (IEEE, 2011).

[60] Forklift Trucks - the Backbone of the Industry. *MHEDA Journal Online* (2004).

[61] Modern Material Handling. Top 20 Lift Truck Suppliers, 2014 (2014). URL `http://www.mmh.com/article/top_20_lift_truck_suppliers_2014`.

[62] JBT Automated Guided Vehicles. Pallet Handling Automatic Guided Vehicles. URL `http://www.jbtc-agv.com/en/Solutions/Applications/Pallet-Handling`.

[63] Dematic. Automated Pallet Transport and/or Storage (AGV). `www.dematic.com/en-US/Supply-Chain-Solutions/By-Vertical-Market/Typical-Solutions/Automate-Pallet-Transport-and-Storage-AGV`.

[64] Automation, E. Egemin Automation - Automated Guided Vehicles and Integrated Material Handling Solutinos. `http://www.egeminusa.com/`.

[65] Seegrid. Seegrid Vision - the Power to Transform. `http://www.seegrid.com/`.

[66] ANSI. *ANSI/ITSDF B56.5-201 2Safety standard for driverless, automatic guided industrial vehicles and automated functions of manned industrial vehicles* (American National Standards Institute, 2012).

[67] Sabattini, L. *et al.* Technological roadmap to boost the introduction of agvs in industrial applications. In *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 203–208 (2013).

[68] Kelly, A., Nagy, B., Stager, D. & Unnikrishnan, R. An Infrastructure-Free Automated Guided Vehicle Based on Computer Vision. *IEEE Robotics and Automation Magazine* (2007).

[69] Sabattini, L. *et al.* Experimental comparison of 3D vision sensors for mobile robot localization for industrial application: Stereo-camera and RGB-D sensor. In *12th International Conference on Control Automation Robotics & Vision, ICARCV 2012, Guangzhou, China*, 823–828 (2012).

[70] Reinke, C. & Beinschob, P. Strategies for Contour-Based Self-Localization in Large-Scale Modern Warehouses. In *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 223–227 (2013).

[71] Beinschob, P. & Reinke, C. Advances in 3d data acquisition, mapping and localization in modern large-scale warehouses. In *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, 265–271 (IEEE, 2014).

[72] Schwesinger, D., Shariati, A., Montella, C. & Spletzer, J. A smart wheelchair ecosystem for autonomous navigation in urban environments. *Autonomous Robots* 1–20 (2016). URL `http://dx.doi.org/10.1007/s10514-016-9549-1`.

[73] Gadd, M. & Newman, P. A framework for infrastructure-free warehouse navigation. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 3271–3278 (IEEE, 2015).

[74] Beinschob, P. & Reinke, C. Graph slam based mapping for agv localization in large-scale warehouses. In *Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on*, 245–248 (IEEE, 2015).

[75] PAN-Robots. URL `http://www.pan-robots.eu`.

[76] Grisetti, G., Stachniss, C. & Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on* **23**, 34–46 (2007).

[77] Censi, A. An icp variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 19–25 (IEEE, 2008).

[78] Velodyne. VLP-16 LiDAR. URL `http://velodynelidar.com/vlp-16.html`.

[79] Grisetti, G., Stachniss, C. & Burgard, W. GMapping. URL `https://www.openslam.org/gmapping.html`.

[80] Smith, R., Self, M. & Cheeseman, P. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, 167–193 (Springer, 1990).

[81] Montella, C., Pollock, M., Schwesinger, D. & Spletzer, J. R. Stochastic classification of urban terrain for smart wheelchair navigation. In *Proceedings of the IROS Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs* (2012).

[82] Couprie, C., Farabet, C., Najman, L. & LeCun, Y. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572* (2013).

[83] Hermans, A., Floros, G. & Leibe, B. Dense 3d semantic mapping of indoor scenes from rgb-d images 2631–2638 (2014).

[84] Konolige, K. & Bowman, J. Towards lifelong visual maps. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 1156–1163 (IEEE, 2009).

[85] Churchill, W. & Newman, P. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 4525–4532 (IEEE, 2012).

[86] Olson, E. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems* **57**, 1157–1172 (2009).

[87] Mühlfellner, P. *et al.* Summary maps for lifelong visual localization. *Journal of Field Robotics* (2015).

[88] Chen, C., Tay, C., Laugier, C. & Mekhnacha, K. Dynamic environment modeling with gridmap: a multiple-object tracking application. In *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*, 1–6 (IEEE, 2006).

[89] Tipaldi, G. D., Meyer-Delius, D. & Burgard, W. Lifelong localization in changing environments. *The International Journal of Robotics Research* **32**, 1662–1678 (2013).

[90] Wang, C.-C., Thorpe, C., Thrun, S., Hebert, M. & Durrant-Whyte, H. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research* **26**, 889–916 (2007).

Vita

Dylan Schwesinger was born in Walton, New York, on March 19, 1979 to Greta Backlund and Frank Schwesinger. He graduated from Honesdale High School in 1997. Between 2007 and 2011 he studied computer science at Kutztown University. He received his B.S. in computer science in from Kutztown University in 2010 and his M.S. in computer science from Kutztown University in 2011.